

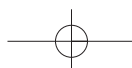
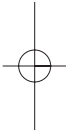


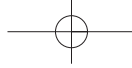
Antworten auf die Wiederholungsfragen



Freitagabend

- 1. Unsere Lösung ist einfacher als das Programm zum Entfernen von Reifen. Wir müssen den Wagenheber nicht erst greifen. Weil das Auto bereits in der Luft ist, gehalten vom Wagenheber, können wir davon ausgehen, dass wir wissen, wo sich der Wagenheber befindet.**
 1. Greife Griff des Wagenhebers
 2. während Auto nicht auf Boden
 3. bewege Griff des Wagenhebers nach unten
 4. bewege Griff des Wagenhebers nach oben
 5. lasse Griff des Wagenhebers los
- 2. Das Entfernen des Minuszeichens zwischen 212 und 32 veranlasst Visual C++ zur fehlerhaften Fehlermeldung »missing ;«.**
- 3. Ich habe das Problem behoben, indem ich ein Semikolon nach 212 eingefügt habe und das Programm neu erzeugt habe. Visual C++ erzeugt das »korrigierte« Programm ohne Beanstandung.**
- 4. Das korrigierte Programm berechnet eine Temperatur in Fahrenheit von 244, was offensichtlich falsch ist.**
- 5. Offensichtlich sind `nFactor = 212;` und `32;` legale Kommandos. Der falsche Ergebniswert von `nFactor` kommt von der fehlerhaften Durchführung der Konvertierung.**
- 6. Ein Minuszeichen zu vergessen, ist ein verständlicher Fehler für einen so schlechten Tipper wie mich. Hätte Visual C++ den Fehler so behoben wie es dachte, dass ein Semikolon fehlt, wäre das Programm ohne Fehler erzeugt worden; die Ausführung liefert jedoch ein falsches Ergebnis. Ich hätte selber durch das Programm gehen müssen, um den Fehler in der Berechnung zu finden. Das schafft Misstrauen in die Umrechnungsformel zwischen Celsius und Fahrenheit, während das eigentliche Problem ein Tippfehler ist. Das ist Zeitverschwendung, wenn doch die Fehlermeldung von Visual C++, wenn auch nicht korrekt, mich direkt zum Ort des Fehlers führt.**





364 Anhang A

7. Erneutes Erzeugen von `Conversion.cpp` nach dem Entfernen der Hochkommata erzeugt diese Fehlermeldung:

`Conversion.cpp(31) Error: unterminated string or character constant`

Diese Fehlermeldung zeigt, dass GNU C++ denkt, dass der Fehler in Zeile 31 aufgetreten ist. (Das ist die Bedeutung von »31« in der Fehlermeldung.)

8. Weil GNU C++ keine Hochkommata gefunden hat, um die Zeichenkette in Zeile 28 zu beenden, dachte es, dass die Zeichenkette bis zu den nächsten Hochkommata geht, was in Zeile 31 der Fall ist. (Diese Hochkommata sind aber tatsächlich der Anfang einer neuen Zeichenkette, aber GNU C++ weiß das nicht.)

Samstagmorgen

1.

- *Teilen von `nTons` durch `1,1` verursacht einen Rundungsfehler.*
- *`2/1,1` ist gleich `1,8`, was auf `1` abgerundet wird. Die Funktion gibt `1000` zurück.*
- *Das Ergebnis der Division von `2` durch `1,1` ist ein `double`-Wert. Die Zuweisung dieses Werts an `nLongTons` resultiert in einer Demotion, die vom Compiler bemerkt werden sollte. Zusatz: »Assign `double` to `int`. Possible loss of significance.« oder etwas in dieser Art.*

2. Die folgende Funktion hat weniger Probleme mit Rundungsfehlern als ihr Vorgänger:

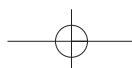
```
int ton2kg(int nTons)
{
    return (nTons * 1000) / 1.1;
}
```

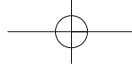
3.

- *5000 Tonnen werden in `4.500.0000.0000` g konvertiert. Diese Zahl liegt außerhalb des Bereiches von `int` auf einem Intel-basierten PC.*
- *Die einzige mögliche Lösung ist die Rückgabe eines `float` oder eines `double` anstelle von `int`. Der Bereich von `float` ist viel größer als der Bereich von `int`.*

4.

- *falsch*
- *wahr*
- *wahr*
- *unbestimmt, aber wahrscheinlich wahr*
Sie können nicht darauf zählen, dass zwei unabhängig voneinander berechnete Gleitkom-mavariablen gleich sind.



**Antworten auf die Wiederholungsfragen 365**

- falsch

Der Wert von `n4` ist gleich 4. Weil die linke Seite von `&&` falsch ist, wird die rechte Seite nicht ausgewertet und `==` wird nie ausgeführt (siehe den Abschnitt über kurze Schaltkreise).

5.

- `0x5D`

- `93`

- `1011 10102`

Es ist am einfachsten, die Addition binär auszuführen. Erinnern Sie sich an die Regeln:

`0 + 0 -> 0`

`1 + 0 -> 1`

`0 + 1 -> 1`

`1 + 1 -> 0, übertrage die 1`

*Alternativ konvertieren Sie die `93 * 2 = 186` zurück ins Binärformat.*

*Zusatz: `0101 11012 * 2` hat das gleiche Bitmuster, nur um eine Position nach links geschiftet, und eine 0 an der Position ganz rechts.*

- `0101 11112`

Konvertiere 2 ins Binärformat `0000 00102` und verknüpfe die beiden Zahlen mit OR.

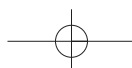
- wahr

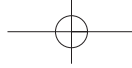
Das Bit `0000 00102` ist nicht gesetzt. Somit ergibt eine AND-Verknüpfung der beiden Null (0).

6. *Erinnern Sie sich daran, dass C++ Leerraum ignoriert, Tabulatoren eingeschlossen. Während der `else`-Zweig scheinbar zum äußeren `if` gehört, gehört es aber tatsächlich zum inneren `if`, so, als wenn es so geschrieben worden wäre:*

```
int n1 = 10;
if (n1 > 11)
{
    if (n1 > 12)
    {
        n1 = 0;
    }
    else
    {
        n1 = 1;
    }
}
```

Die äußere `if`-Anweisung hat keinen `else`-Zweig, und `n1` bleibt daher unverändert.



**366 Anhang A**

7. Weil $n1$ nicht kleiner als 5 ist, wird der Body von `while()` nie ausgeführt. Im Falle von `do...while()` wird der Body einmal ausgeführt, obwohl $n1$ nicht kleiner als 5. In diesem Fall erhält $n1$ den Wert 11.
Der Unterschied zwischen den beiden Schleifen ist, dass `do...while()` seinen Body immer mindestens einmal ausführt, selbst wenn die Bedingung gleich zu Beginn falsch ist.

8.

```
double cube(double d)
{
    return d * d * d;
}
```

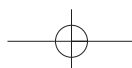
Weil der Intel-Prozessor in Ihrem PC Integerzahlen und Gleitkommazahlen unterschiedlich behandelt, ist der Maschinencode, der von den Funktionen `cube(int)` und `cube(double)` erzeugt wird, voneinander verschieden.

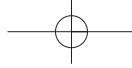
9. Der Ausdruck `cube(3.0)` passt zu der Funktion `cube(double)`; somit wird `cube(double)` der Wert 3.0 übergeben, die den Wert 9.0 zurückgibt, der zu 9 demotiert wird und der Variablen n zugewiesen wird. Obwohl das Ergebnis das gleiche ist, ist der Weg dorthin verschieden.
10. Der Compiler erzeugt einen Fehler, weil die erste Funktion `cube(int)` und diese neue Funktion den gleichen Namen haben. Erinnern Sie sich daran, dass der Rückgabotyp nicht Teil des Namens ist.

Samstagnachmittag

1.

```
class Student
{
    public:
        char szLastName[128];
        int nGrade; // 1-> 1. Grad, 2-> 2. Grad ...
        double dAverage;
}
```



**Antworten auf die Wiederholungsfragen 367****2.**

```
void readAndDisplay()
{
    Student s;

    // Eingabe Studenteninformaton
    cout << »Name des Studenten:<<;
    cin.getline(s.szLastName);
    cout << »Grad (1, 2, ...)\\n<<;
    cin >> s.nGrade;
    cout << »Durchschnitt:<<;
    cin >> s.dGPA;

    // Ausgabe der Studenteninformaton
    cout << »\\nStudenteninformaton:\\n<<;
    cout << s.szLastName << »\\n<<;
    cout << s.nGrade << »\\n<<;
    cout << s.dAverage << »\\n<<;
}
```

3.

```
void readAndDisplayAverage()
{
    Student s;

    // Eingabe Studenteninformaton
    cout << »Name des Stundenten:<<;
    cin.getline(s.szLastName);
    cout << »Grad (1, 2, ...)\\n<<;
    cin >> s.nGrade;
    cout << »Durchschnitt:<<;

    // drei Grade, die gemittelt werden
    double dGrades[3];
    cin >> dGrade[0];
    cin >> dGrade[1];
    cin >> dGrade[2];
    s.dAverage = (dGrade[0] + dGrade[1] + dGrade[2]) / 3;

    // Ausgabe der Studenteninformaton
    cout << »\\nStudenteninformaton:\\n<<;
    cout << s.szLastName << »\\n<<;
    cout << s.nGrade << »\\n<<;
    cout << s.dAverage << »\\n<<;
}
```

368 **Anhang A**

4.

- 16 Bytes ($4 + 4 + 4$)
- 80 Bytes ($4 * 20$)
- 8 ($4 + 4$) *Erinnern Sie sich daran, dass die Größe eines Zeigers 4 Bytes ist, unabhängig davon, auf was der Zeiger zeigt.*

5.

- Ja.
- *Sie alloziert Speicher vom Heap, gibt diesen aber nicht zurück, bevor die Funktion verlassen wird (so etwas wird als Speicherloch bezeichnet).*
- *Bei jedem Aufruf geht ein wenig Speicher verloren, bis der Heap aufgebraucht ist.*
- *Es kann sehr lange dauern, bis der Speicher aufgebraucht ist. Bei einem solch kleinen Speicherloch muss das Programm viele Stunden laufen, nur damit das Problem überhaupt sichtbar wird.*

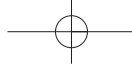
6. `dArray[0]` ist bei `0x100`, `dArray[1]` ist bei `0x108` und `dArray[2]` ist bei `0x110`. Das Array erstreckt sich von `0x100` bis `0x118`.

7. Zuweisung 1 hat den gleichen Effekt wie `dArray[1] = 1,0`. Die Zuweisung 2 zerstört den Gleitkommawert in `dArray[2]`, das ist aber nicht fatal, weil 4 Bytes, die für `int` benötigt werden, in die 8 Bytes, die für `double` alloziert wurden, hineinpassen.

8.

```
LinkableClass* removeHead()
{
    LinkableClass* pFirstEntry;
    pFirstEntry = pHead;
    if (pHead != 0)
    {
        pHead = pHead->pNext;
    }
    return pFirstEntry;
}
```

Die Funktion `removeHead()` überprüft erst, ob der Kopfzeiger null ist. Wenn dies der Fall ist, ist die Liste bereits leer. Wenn nicht, speichert die Funktion einen Zeiger auf das erste Element in `pFirstEntry`. Sie bewegt dann `pHead` ein Element weiter, und gibt schließlich `pFirstEntry` zurück.

**Antworten auf die Wiederholungsfragen 369**

9.

```
LinkableClass* returnPrevious(LinkableClass* pTarget)
{
    // gib null zurück, wenn die Liste leer ist
    if (pHead == 0)
    {
        return 0;
    }

    // jetzt iteriere durch die Liste
    LinkableClass* pCurrent= pHead;
    while(pCurrent->pNext)
    {
        // wenn der pNext-Zeiger des aktuellen
        // Eintrags gleich pTarget ist...
        if (pCurrent->pNext == pTarget)
        {
            // ... dann gib pCurrent zurück
            return pCurrent;
        }
    }

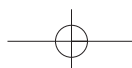
    // wenn wir durch die gesamte Liste durch sind,
    // ohne pTarget zu finden, gib null zurück
    return 0;
}
```

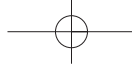
Die Funktion `returnPrevious()` **gibt den Eintrag in der Liste zurück, der unmittelbar vor *pTarget steht. Die Funktion beginnt damit, zu überprüfen, ob die Liste leer ist. Wenn sie leer ist, gibt es keinen Vorgängereintrag, und die Funktion gibt null zurück.**

`returnPrevious()` **iteriert dann durch die Liste, wobei pCurrent auf den aktuellen Listeneintrag zeigt. In jedem Schleifendurchlauf überprüft die Funktion, ob das nächste Element von pCurrent aus gleich pTarget ist. Ist dies der Fall, gibt die Funktion pCurrent zurück. Wenn returnPrevious() durch die gesamte Liste durchgegangen ist, ohne pTarget zu finden, gibt die Funktion null zurück.**

10.

```
LinkableClass* returnTail()
{
    // gib den letzten Eintrag der
    // Liste zurück; das ist der Eintrag,
    // dessen Nachfolger null ist
    return returnPrevious(0);
}
```



**370 Anhang A**

Der Eintrag, auf den null folgt, ist der letzte Eintrag in der Liste.

Zusatzaufgabe:

```
LinkableClass* removeTail()
{
    // finde den letzten Eintrag der Liste; wenn er
    // null ist, dann ist die Liste leer
    LinkableClass* pLast = returnPrevious(0);
    if (pLast == 0)
    {
        return 0;
    }

    // jetzt finde den Eintrag, der auf diesen
    // letzten Eintrag verweist
    LinkableClass* pPrevious = returnPrevious(pLast);

    // wenn pPrevious null ist ...
    if (pPrevious == 0)
    {
        // ... dann ist pLast der einzige Eintrag;
        // setze den Kopfzeiger auf null
        pHead = 0;
    }
    else
    {
        // ... sonst entferne pLast aus pPrevious
        pPrevious->pNext = 0;
    }

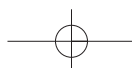
    // in jedem Fall gib den Zeiger pLast zurück
    return pLast;
}
```

Die Funktion `removeTail()` **entfernt das letzte Element in einer verketteten Liste. Sie beginnt damit, die Adresse des letzten Elements durch einen Aufruf von** `returnPrevious(0)` **zu bestimmen. Sie speichert diese Adresse in** `pLast`. **Wenn** `pLast` **gleich null ist, dann ist die Liste leer und die Funktion gibt sofort null zurück.**

Das letzte Element zu finden reicht nicht aus. Um das letzte Element zu entfernen, muss `removeTail()` **den Eintrag vor** `pLast` **finden, damit die beiden getrennt werden können.**

`removeTail()` **bestimmt die Adresse des Eintrages von** `pLast` **durch einen Aufruf** `returnPrevious(pLast)`. **Wenn es keinen solchen Eintrag gibt, enthält die Liste nur ein Element.**

`removeTail()` **setzt den entsprechenden Zeiger auf null und gibt dann** `pLast` **zurück.**



Antworten auf die Wiederholungsfragen 371

11. Ich habe für diese Lösung Visual C++ verwendet, um einen Vergleich zu `rhide` zu bekommen.

Ich beginne damit, das Programm auszuführen, einfach um zu sehen, was passiert. Wenn das Programm funktioniert, dann ist es ja gut. Das Ergebnis bei Eingabe von »diese Zeichenkette« und »DIESE ZEICHENKETTE« sieht wie folgt aus:

Dieses Programm verbindet zwei Zeichenketten:
(Diese Version stürzt ab)

```
Zeichenkette #1:diese Zeichenkette
Zeichenkette #2:DIESE ZEICHENKETTE
```

```
DIE
Press any key to continue
```

Weil ich mir sicher bin, dass das Problem in `concatString()` liegt, setzte ich einen Haltepunkt auf den Anfang der Funktion, und starte erneut. Nachdem der Haltepunkt angetroffen wurde, scheinen die beiden Zeichenketten, Quelle und Ziel, korrekt zu sein, wie sie in Abbildung 1.1 sehen können.

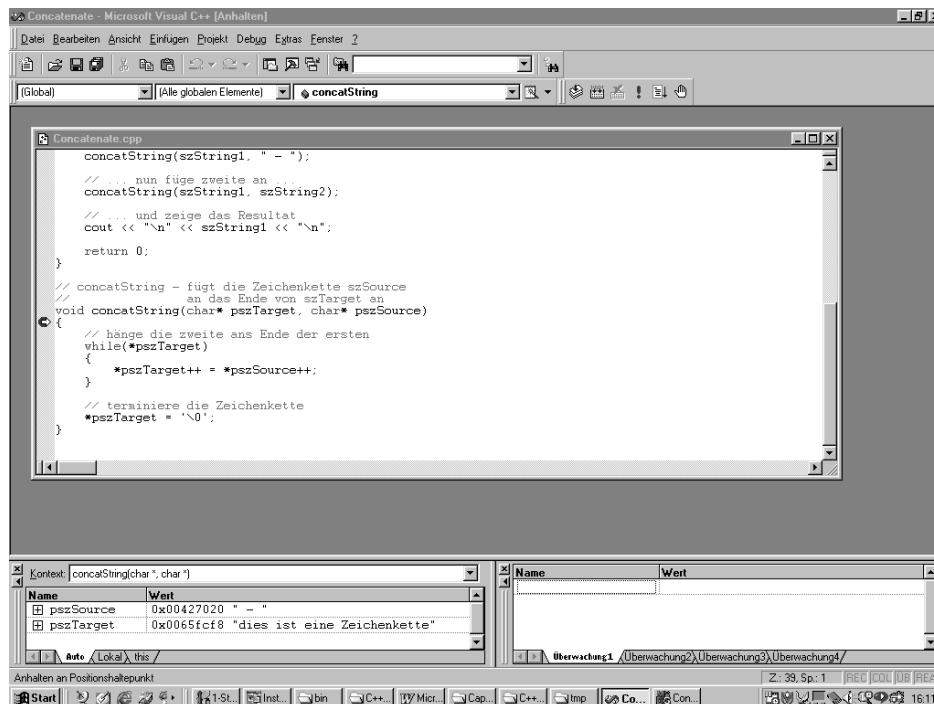
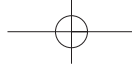


Abbildung A.1: Das Fenster zeigt die Quellzeichenkette und die Zielzeichenkette an.



372 Anhang A

Ausgehend vom Haltepunkt gehe ich in Einzelschritten vor. Zuerst scheinen die lokalen Variablen in Ordnung zu sein. Sobald jedoch die `while`-Schleife betreten wird, wird sofort klar, dass die Quellzeichenkette die Zielzeichenkette überschreibt; die Funktion `concatString()` ist eher eine Überschreibefunktion.

Die Funktion `concatString()` sollte damit anfangen, `pszTarget` an das Ende der Zeichenkette zu bewegen, bevor der Kopierprozess beginnt. Das Problem lässt sich nicht so einfach im Debugger beheben, und ich füge den Extracode in das Programm ein.



Tatsächlich ist es möglich, dieses Problem mit Hilfe des Debuggers zu lösen. In der Anzeige der lokalen Variablen, klicken Sie auf den Wert des Zeigers in der Spalte rechts neben `pszTarget`. Was immer dort für ein Wert steht, addieren Sie `0x12` (es gibt 18 Zeichen in »diese Zeichenkette«). Der Zeiger `pszTarget` zeigt jetzt auf das Ende der Zielzeichenkette, und das Kopieren der Zeichen kann beginnen.

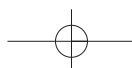
Die geänderte Funktion `concatString()` sieht wie folgt aus:

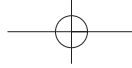
```
void concatString(char* pszTarget, char* pszSource)
{
    // bewege pszTarget ans Ende der Zielzeichenkette
    while(*pszTarget)
    {
        pszTarget++;
    }

    // füge die zweite ans Ende der ersten an
    while(*pszTarget)
    {
        *pszTarget++ = *pszSource++;
    }

    // terminate the string properly
    *pszTarget = '\0';
}
```

Ich setze den Haltepunkt auf das zweite `while`, das ist die Schleife, die das Kopieren ausführt, und starte das Programm erneut mit der gleichen Eingabe. Die lokalen Variablen sind korrekt, wie in Abbildung A.2 zu sehen ist.





Antworten auf die Wiederholungsfragen 373



Abbildung A.2: *pszTarget* sollte auf eine Null zeigen, bevor das Kopieren durchgeführt wird.



Hinweis

Die Variable *pszTarget* sollten auf die Null am Ende der Zielzeichenkette zeigen, bevor das Programm die Quellzeichenkette kopieren kann.

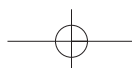
Voller Vertrauen, versuche ich, durch die `while`-Schleife hindurchzugehen. Sobald ich jedoch »Step Over« drücke, überspringt das Programm die Schleife. Offensichtlich ist die `while`-Bedingung selbst zu Beginn nicht wahr. Nach kurzem Nachdenken stelle ich fest, dass die Bedingung falsch ist. Anstatt anzuhalten, wenn *pszTarget* gleich null ist, sollte ich anhalten, wenn *pszSource* gleich null ist. Umschreiben der `while`-Schleife, wie folgt, löst das Problem:

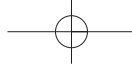
```
while(*pszSource)
```

Ich starte das Programm erneut mit der gleichen Eingabe, und der Debugger zeigt an, dass alles in Ordnung zu sein scheint. Und das Programm erzeugt auch die richtige Ausgabe.

Samstagabend

1. Ich finde Hemden und Hosen, die Unterklassen von Kleidungsstück sind, was Unterklasse von Bekleidung ist. In der Reihe der Bekleidungen stehen auch Schuhe und Sockenpaare. Die Sockenpaare können weiter unterteilt werden in die Paare, die zusammenpassen, und in die Paare, die nicht passen, usw.
2. Schuhe haben zumindest eine Öffnung, um den Fuß hineinzustecken. Schuhe haben eine Art Befestigungssystem, um sie am Fuß zu halten. Schuhe haben eine Sohle, um sie gegen den Untergrund zu schützen. Das ist alles, was ich über meine Schuhe sagen kann.
3. Ich habe Anzugsschuhe und Fahrradschuhe. Ich kann meine Fahrradschuhe zur Arbeit anziehen, es ist aber sehr schwer, darin zu laufen. Sie umschließen jedoch die Füße, und die Arbeit würde dadurch nicht zum Erliegen kommen.





374 Anhang A

So nebenbei – ich habe auch ein paar Kombinationsschuhe, die das Verbindungselement zur Pedale in die Sohle eingelassen haben. In diesen Schuhen zur Arbeit zu gehen wäre nicht ganz so schlecht.

4. Der Konstruktor eines Objektes der verketteten Liste muss sicherstellen, dass der Zeiger auf das nächste Element auf null zeigt, wenn das Objekt konstruiert wird. Es ist nicht notwendig, etwas mit den statischen Elementen zu tun.

```
class Link
{
    static Link* pHead;
    Link* pNextLink;

    Link()
    {
        pNextLink = 0;
    }
};
```

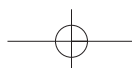
Der Punkt ist, dass das statische Element pHead nicht im Konstruktor initialisiert werden kann, weil es sonst jedes Mal initialisiert wird, wenn ein neues Objekt erzeugt wird.

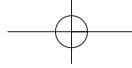
5. Dies ist meine Version des Destruktors und des Kopierkonstruktors:

```
// Destruktor - vernichte Objekt und Eintrag
~LinkedList()
{
    // wenn aktuelle Objekt in der Liste ist ...
    if (pNext)
    {
        // ... dann entferne es
        removeFromList();
    }

    // wenn das Objekt Speicher belegt ...
    if (pszName)
    {
        // ... gib ihn an den Heap zurück
        delete pszName;
    }
    pszName = 0;
}

// Kopierkonstruktor - macht eine Kopie eines
// existierenden Objektes
LinkedList(LinkedList& l)
{
    // alloziere einen Block der gleichen
    // Größe vom Heap
    int nLength = strlen(l.pszName) + 1;
    this->pszName = new char[nLength];
```





Antworten auf die Wiederholungsfragen 375

```
// kopiere den Namen in diesen Block
strcpy(this->pszName, l.pszName);

// setze Zeiger auf null, da Element nicht mehr
// in verketteter Liste ist
pNext = 0;
}
```

Wenn das Objekt in einer verketteten Liste steht, dann muss der Destruktor es entfernen, bevor das Objekt wiederverwendet und der Zeiger pNext verloren ist. Wenn das Objekt zusätzlich einen Speicherbereich »besitzt«, muss dieser zurückgegeben werden. In gleicher Weise führt der Kopierkonstruktor eine tiefe Kopie aus, indem er Speicher vom Heap alloziert, in dem der Name gespeichert wird. Der Kopierkonstruktor fügt das Objekt nicht in die existierende Liste ein (obwohl er das könnte).

Sonntagmorgen

1. Die Ausgabe lautet:

```
Advisor:Student Datenelement
Student
Advisor:Student lokal
Advisor:GraduateStudent Datenelement
GraduateStudent
Advisor: GraduateStudent lokal
```

Lassen Sie uns jede Zeile der Ausgabe ansehen:

Die Kontrolle geht an den Konstruktor von GraduateStudent und von dort an den Konstruktor der Basisklasse Student über.

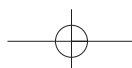
Das Datenelement Student::adv wird konstruiert.

Die Kontrolle geht an den Body des Konstruktors Student über.

Ein lokales Objekt aus der Klasse Advisor wird vom Heap erzeugt.

Die Kontrolle kehrt zum Konstruktor GraduateStudent zurück, der das Datenelement GraduateStudent::adv erzeugt.

Die Kontrolle betritt den Konstruktor GraduateStudent, der ein Advisor-Objekt vom Heap alloziert.



376 **Anhang A****2.**

```
// PassProblem - nutze Polymorphie, um zu
//                entscheiden, ob ein Student
//                besteht oder durchfällt
#include <stdio.h>
#include <iostream.h>

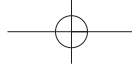
class Student
{
public:
    virtual int pass(double dGrade)
    {
        // wenn es zum Bestehen reicht ...
        if (dGrade > 1.5)
        {
            // ... bestanden
            return 1;
        }
        // ... sonst durchgefallen
        return 0;
    }
};

class GraduateStudent : public Student
{
public:
    virtual int pass(double dGrade)
    {
        if (dGrade > 2.5)
        {
            return 1;
        }
        return 0;
    }
};
```

3. Meine Version der Klasse Checking ist wie folgt:

```
// Checking - gleich mit Sparkonten, außer dass
//                Abhebungen einen Dollar kosten
class Checking : public CashAccount
{
public:
    Checking(unsigned nAccNo,
             float fInitialBalance = 0.0F)
        : CashAccount(nAccNo, fInitialBalance)
    {
    }

    // ein Girokonto weiß, wie Abhebungen
    // durchgeführt werden (machen Sie sich keine
    // Gedanken zu Überziehungen)
    virtual void withdrawal(float fAmount)
    {
```

**Antworten auf die Wiederholungsfragen 377**

```
        // Abhebung ausführen
        fBalance -= fAmount;

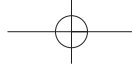
        // Gebühr erheben
        fBalance -= 1;
    }
};
```



Das gesamte Programm ist auf der beiliegenden CD-ROM enthalten unter dem Namen *AbstractProblem*.

4. Das ist meine Lösung des Problems:

```
// MultipleVirtual - erzeuge Klasse PrinterCopier
//                      durch Erben von Printer und
//                      Copier
#include <stdio.h>
#include <iostream.h>
class ElectronicEquipment
{
public:
    ElectronicEquipment(int nVoltage)
    {
    }
    int nVoltage;
};
class Printer : virtual public ElectronicEquipment
{
public:
    Printer() : ElectronicEquipment()
    {
    }
    void print()
    {
        cout << »drucken\n«;
    }
};
class Copier : virtual public ElectronicEquipment
{
public:
    Copier() : ElectronicEquipment()
    {
    }
    void copy()
    {
        cout << »kopieren\n«;
    }
};
class PrinterCopier : public Printer, public Copier
{
```

**378 Anhang A**

```
public:
    PrinterCopier(int nVoltage) : Copier(), Printer()
    {
        this->nVoltage = nVoltage;
    }
};

int main(int nArgs, char* pszArgs)
{
    PrinterCopier ss(220);

    // erst drucken
    ss.print();

    // dann kopieren
    ss.copy();

    // Spannung ausgeben
    cout << »Spannung = » << ss.nVoltage << »\n«;

    return 0;
}
```

Sonntagnachmittag**1. Meine Version der beiden Funktionen sieht wie folgt aus:**

```
MyClass(MyClass& mc)
{
    nValue = mc.nValue;
    resource.open(nValue);
}
MyClass& operator=(MyClass& s)
{
    resource.close();

    nValue = s.nValue;
    resource.open(nValue);
}
```

Der Kopierkonstruktor öffnet das aktuelle Objekt mit dem Wert des Quellobjektes.

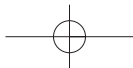
Der Zuweisungsoperator schließt zuerst die aktuelle Ressource, weil sie mit einem anderen Wert geöffnet wurde. Sie öffnet dann die Ressource wieder mit dem neuen Wert, der übergeben wurde.

2. Meine Klasse und Inserter sind wie folgt:

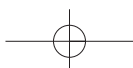
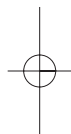
```
// Student
class Student
{
    friend ostream& operator<<(ostream& out, Student& d);
public:
    Student(char* pszFName, char* pszLName, int nSSNum)
    {
        strncpy(szFName, pszFName, 20);
        strncpy(szLName, pszLName, 20);
        this->nSSNum = nSSNum;
    }

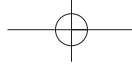
protected:
    char szLName[20];
    char szFName[20];
    int nSSNum;
};

// Inserter - Zeichenkettenbeschreibung
ostream& operator<<(ostream& out, Student& s)
{
    out << s.szLName
        << », »
        << s.szFName
        << »(»
        << s.nSSNum
        << »)<<;
    return out;
}
```



vakat





Ergänzende Probleme



Dieser Anhang enthält ergänzende Probleme für verschiedene Teile des Buches, die Ihnen zusätzliche Erfahrung bei der Arbeit mit C++ bringen und ihre neuen Fähigkeiten festigen sollen. Die Probleme finden Sie in Kapiteln, die den einzelnen Buchteilen zugeordnet sind, jeweils gefolgt von einem Abschnitt mit Lösungen für die Probleme.

1.1 Probleme

1.1.1 Samstagmorgen

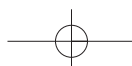
1. Welche der folgenden Anweisungen sollte

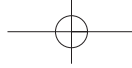
- a. keine Meldungen erzeugen?
- b. Warnungen erzeugen?
- c. Fehler erzeugen?

```
1. int n1, n2, n3;  
2. float f1, f2 = 1;  
3. double d1, d2;  
4. n1 = 1; n2 = 2; n3 = 3; f2 = 1;  
5. d1 = n1 * 2.3;  
6. n2 = n1 * 2.3;  
7. n2 = d1 * 1;  
8. n3 = 100; n3 = n3 * 1000000000;  
9. f1 = 200 * f2;
```

2. Gegeben, dass $n1$ gleich 10 ist, werten sie das Folgende aus:

- a. $n1 / 3$
- b. $n1 \% 3$
- c. $n1++$
- d. $++n1$
- e. $n1 \% = 3$
- f. $n1 -= n1$
- g. $n1 = -10$; $n1 = +n1$; was ist $n1$?




382 Anhang B
3. Was ist der Unterschied der beiden folgenden for-Schleifen:

```
for(int i = 0; i < 10; i++)
{
    // ...
}
for (int i = 0; i < 10; ++i)
{
    // ...
}
```

4. Schreiben Sie die Funktion `int cube(int)`, die $n * n * n$ für n berechnet.
5. Beschreiben Sie genau, was passiert, wenn die Funktion aus Problem 4 wie folgt verwendet wird:

```
int n = cube(3.0);
```

6. Das folgende Programm zeigt eine sehr wenig durchdachte Funktion, die die Integerquadratwurzel einer Zahl berechnet, durch Vergleiche mit dem Quadrat eines Zählers. Mit anderen Worten, es ist $4 * 4 = 16$, woraus folgt, dass 4 die Quadratwurzel von 16 ist. Diese Funktion berechnet 3 als Quadratwurzel von 15, weil $3 * 3$ kleiner ist als 15, aber $4 * 4$ größer ist als 15.

Das Programm erzeugt jedoch unerwartete Ergebnisse. Beheben Sie den Fehler!

```
// Lesson - diese Funktion zeigt eine wenig
// durchdachte, aber effektive Methode
// zur Berechnung der Quadratwurzel, wobei
// nur Integers verwendet werden.
// Diese Version funktioniert nicht.
#include <stdio.h>
#include <iostream.h>

// squareRoot - gegeben eine Zahl n, gib ihre
// Quadratwurzel zurück, indem
// nRoot * nRoot für aufsteigende Werte
// nRoot berechnet wird, bis das Quadrat
// größer ist als n
void squareRoot(int n)
{
    // starte bei 1
    int nRoot = 1;

    // Endlosschleife
    for(;;)
    {
        // überprüfe Quadrat des aktuellen Werts
        // (und inkrementiere zum nächsten)
        if ((nRoot++ * nRoot) > n)
```

```

    {
        // so nah wie möglich, gib diesen
        // Wert zurück
        return nRoot;
    }

    // hier sollten wir nicht hinkommen
    return 0;
}

// teste die Funktion squareRoot() mit einem
// einzelnen Wert (ein ausführlicherer Test
// wäre angebracht, aber schon dieser Test
// funktioniert nicht)
int main(int argc, char* pszArgs[])
{
    cout << »Dieses Programm funktioniert nicht!\n<<;

    cout << »Quadratwurzel von »
        << 16
        << » ist »
        << squareRoot(16)
        << »\n<<;
    return 0;
}

```

Hinweis: Achten Sie auf die Features Autoinkrement und Postinkrement.

1.1.2 Samstagnachmittag

1. Die C++-Bibliotheksfunktion `strchr()` gibt den Index eines Zeichens in einer Zeichenkette zurück. So gibt z.B. `strchr(»abcdef«, 'c')` den Index 2 zurück. `strchr()` gibt -1 zurück, wenn das Zeichen in der Zeichenkette nicht vorkommt.

Schreiben Sie eine Funktion `myStrchr()` die das Gleiche tut wie `strchr()`. Wenn Sie denken, dass Ihre Funktion fertig ist, testen Sie sie mit dem Folgenden:

```

// MyStrchr - suche ein gegebenes Zeichen in einer
//           Zeichenkette. Gib den Index des
//           zurück
#include <stdio.h>
#include <iostream.h>

// myStrchr - gib den Index eines Zeichens in einer
//           Zeichenkette zurück; gib -1 zurück,
//           wenn das Zeichen nicht gefunden wird
int myStrchr(char target[], char testChar);

// teste die Funktion myStrchr mit verschiedenen
// Kombinationen von Zeichenketten

```

384 Anhang B

```
void testFn(char szString[], char cTestChar)
{
    cout << »Der Offset von »
         << cTestChar
         << » in »
         << szString
         << » ist »
         << myStrchr(szString, cTestChar)
         << »\n«;
}

int main(int nArgs, char* pszArgs[])
{
    testFn(»abcdefg«, 'c');
    testFn(»abcdefg«, 'a');
    testFn(»abcdefg«, 'g');
    testFn(»abcdefc«, 'c');
    testFn(»abcdefg«, 'x');

    return 0;
}
```

Hinweis: Achten Sie darauf, dass Sie nicht aus der Zeichenkette herauslaufen, wenn das Zeichen nicht gefunden wird.

2. Obwohl das Folgende schlecht programmiert ist, verursacht es doch keine Probleme. Erklären Sie warum nicht.

```
void fn(void)
{
    double d;
    int* pnVar = (int*)&d;
    *pnVar = 10;
}
```

3. Schreiben Sie eine Zeigerversion der folgenden Funktion `displayString()`. Nehmen Sie an, dass das eine Null-terminierte Zeichenkette ist (übergeben Sie nicht die Länge als Argument an die Funktion).

```
void displayCharArray(char sArray[], int nSize)
{
    for(int i = 0; i < nSize; i++)
    {
        cout << sArray[i];
    }
}
```

4. Kompilieren Sie das folgende Programm, und führen Sie es aus. Erklären Sie die Ergebnisse:

```
#include <stdio.h>
#include <iostream.h>
// MyClass - Testklasse ohne Bedeutung
class MyClass
{
public:
    int n1;
    int n2;
};

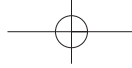
int main(int nArg, char* nArgs[])
{
    MyClass* pmc;
    cout << »n1 = » << pmc->n1
         << »;n2 = » << pmc->n2
         << »\n«;
    return 0;
}
```

1.1.3 Sonntagmorgen

1. **Schreiben Sie eine Klasse `Car`, die von der Klasse `Vehicle` erbt, und die einen Motor besitzt. Die Anzahl der Reifen wird im Konstruktor der Klasse `Vehicle` angegeben, und die Anzahl der Zylinder im Konstruktor von `Motor`. Beide Werte werden an den Konstruktor der Klasse `Car` übergeben.**

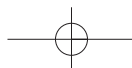
1.2 Antworten**1.2.1 Samstagmorgen**

1. **1. Kein Problem.**
 2. **Keine Warnung; sie können initialisieren, wenn sie wollen.**
 3. **Alles klar.**
 4. **Kein Problem.**
 5. **Kein Problem; $n1$ wird automatisch in ein `double` verwandelt, um die Multiplikation ausführen zu können. Die meisten Compiler werden diese Konvertierung nicht kommentieren.**
 6. **$n1 * 2.3$ ist ein `double`. Die Zuweisung an eine `int`-Variable führt zu einer Warnung wegen Demotion.**


386 Anhang B

7. Ähnlich wie 6. Obwohl 1 ein `int` ist, ist `d1` ein `double`. Das Ergebnis ist ein `double`, das nach `int` konvertiert wird (Demotion).
8. Keine Warnung, aber es funktioniert nicht. Das Ergebnis liegt außerhalb des Bereiches von `int`.
9. Das sollte eine Warnung erzeugen (Visual C++ tut es auch, GNU C++ tut es nicht). Das Ergebnis einer Multiplikation von `int` und `float` ist `double` (alle Berechnungen werden in `double` ausgeführt). Das Ergebnis muss nach `float` konvertiert werden (Demotion).
2. a. 3 – Rundungsfehler, die in Sitzung 5 erklärt wurden, konvertieren das erwartete Ergebnis 3.3 nach 3.
- b. 1 – Der Teiler, der 10 / 3 am nächsten ist, ist 3. 10 – (3 * 3) ist 1, der Rest der Division.
- c. 10 – `n1++` gibt den Wert von `n1` zurück, bevor `n1` inkrementiert wird. Nach der Auswertung des Ausdrucks ist `n1 = 11`.
- d. 11 – `++n1` inkrementiert `n1`, bevor der Wert zurückgegeben wird.
- e. 1 – Das ist das Gleiche wie `n1 = n1 % 3`
- f. 0 – Das ist das Gleiche wie `n1 = n1 - n1`, aber `n1 - n1` ist immer Null.
- g. -10 – Der unäre Plusoperator (+) hat keinen Effekt. Insbesondere ändert er nicht das Vorzeichen einer negativen Zahl.
3. Kein Unterschied. Die Inkrementklausel einer `if`-Anweisung wird als separater Ausdruck behandelt. Der Wert von `i` ist nach einem Präinkrement und nach einem Postinkrement gleich (nur der Wert des Ausdrucks ist verschieden).
4.

```
int cube(int n)
{
    return n * n * n;
}
```
5. Der `double`-Wert 3.0 wird demotiert in den `int`-Wert 3, und das Ergebnis wird als Integerwert 9 zurückgegeben.
6. Fehler #1: Das Programm kann nicht kompiliert werden, weil die Funktion `squareRoot()` so deklariert wurde, dass sie `void` als Rückgabewert hat. Ändern Sie den Rückgabewert auf `int` und erzeugen Sie das Programm erneut.
- Fehler #2: Das Programm behauptet nun, dass die Quadratwurzel von 16 gleich 6 ist. Um das Problem zu verstehen, splitte ich die zusammengesetzte `if`-Bedingung, damit ich den Ergebniswert jeweils ausgeben kann:



```

// Endlosschleife
for(;;)
{
    // überprüfe Quadrat des aktuellen Wertes
    // (und inkrementiere zum nächsten)
    int nTest = nRoot++ * nRoot;
    cout << »Testwurzel ist »
        << nRoot
        << » Quadrat ist »
        << nTest
        << »\n«;
    if (nTest > n)
    {
        // so nah wie möglich, gib diesen
        // Wert zurück
        return nRoot;
    }
}

```

Die Ausgabe des Programms sieht wie folgt aus:

```

Dieses Programm funktioniert nicht!
Testwurzel ist 2 Quadrat ist 1
Testwurzel ist 3 Quadrat ist 4
Testwurzel ist 4 Quadrat ist 9
Testwurzel ist 5 Quadrat ist 16
Testwurzel ist 6 Quadrat ist 25
Quadratwurzel von 16 ist 6

```

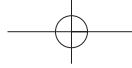
Die Ausgabe des Programms ist überhaupt nicht korrekt. Eine genaue Untersuchung zeigt jedoch, dass die linke Seite um eins verschoben ist. Das Quadrat von 3 ist 9, aber der angezeigte Wert von `nRoot` ist 4. Das Quadrat von 4 ist 16, aber der angezeigte Wert von `nRoot` ist 5. Durch die Inkrementierung von `nRoot` im Ausdruck, ist `nRoot` um eins größer als das `nRoot`, das in der Berechnung verwendet wird. Somit muss `nRoot` nach der `if`-Anweisung inkrementiert werden.

Die neue Funktion `squareRoot()` sieht so aus:

```

// Endlosschleife
for(;;)
{
    // überprüfe Quadrat des aktuellen Wertes
    int nTest = nRoot * nRoot;
    cout << »Testwurzel ist »
        << nRoot
        << » Quadrat ist »
        << nTest
        << »\n«;
    if (nTest > n)
    {
        // so nah wie möglich, gib diesen

```


388 Anhang B

```

        // Wert zurück
        return nRoot;
    }
    // versuche nächsten Wert für nRoot
    nRoot++;
}

```

Das Autoinkrement wurde hinter den Test platziert (das Autoinkrement war die ganze Zeit verdächtig). Die Ausgabe des neuen, verbesserten Programms sieht wie folgt aus:

```

Dieses Programm funktioniert nicht!
Testwurzel ist 1 Quadrat ist 1
Testwurzel ist 2 Quadrat ist 4
Testwurzel ist 3 Quadrat ist 9
Testwurzel ist 4 Quadrat ist 16
Testwurzel ist 5 Quadrat ist 25
Quadratwurzel von 16 ist 5

```

Das Quadrat wird korrekt berechnet, aber aus irgendeinem Grund stoppt die Funktion nicht, wenn `nRoot` gleich 4 ist. Es gilt aber doch $4 * 4 == 16$. Das ist genau das Problem – der Test überprüft `nTest > n`, wo er eigentlich `nTest >= n` überprüfen sollte. Das korrigierte Programm erzeugt die erwartete Ausgabe:

```

Dieses Programm funktioniert!
Testwurzel ist 1 Quadrat ist 1
Testwurzel ist 2 Quadrat ist 4
Testwurzel ist 3 Quadrat ist 9
Testwurzel ist 4 Quadrat ist 16
Quadratwurzel von 16 ist 4

```

Nachdem ich verschiedene Werte getestet habe, bin ich davon überzeugt, dass das Programm korrekt ist, und ich entferne die Ausgabeanweisungen.

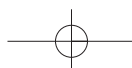
1.2.2 Samstagnachmittag

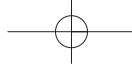
1. Die folgende Funktion `myStrchr()` ist meine Lösung des Problems:

```

// myStrchr - gib den Index eines Zeichens in einer
//             Zeichenkette zurück; gib -1 zurück,
//             wenn das Zeichen nicht gefunden wird
int myStrchr(char target[], char testChar)
{
    // Schleife über alle Zeichen der Zeichenkette;
    // spätestens am Ende der Zeichenkette stoppen
    int index = 0;
    while(target[index])
    {
        // wenn das aktuelle Zeichen der

```





```

// Zeichenkette gleich dem gesuchten
// Zeichen ist ...
if (target[index] == testChar)
{
    // ... dann stoppen
    break;
}

// gehe zum nächsten Zeichen
index++;
}

// wenn wir am Ende der Zeichenkette ankommen,
// ohne das Zeichen zu finden, ...
if (target[index] == '\0')
{
    // ... gib -1 und nicht die Länge
    // des Arrays zurück
    index = -1;
}

// gib den berechneten Index zurück
return index;
}

```

Die Funktion `myStrchr()` beginnt damit, durch die Zeichenkette `target` zu iterieren, wobei gestoppt wird, wenn das aktuelle Zeichen `target[index]` gleich 0 ist, was bedeutet, dass die Funktion das Ende der Zeichenkette erreicht hat. Dieser Test stellt sicher, dass die Funktion nicht zu weit geht, wenn das Zeichen nicht gefunden wird.

Innerhalb dieser Schleife vergleicht die Funktion das aktuelle Zeichen mit dem gesuchten Zeichen `testChar`. Wenn das Zeichen gefunden wird, verläßt die Funktion die Schleife vorzeitig.

Wenn die Schleife verlassen worden ist, wurde entweder das Ende der Zeichenkette angetroffen oder das zu suchende Zeichen wurde gefunden. Wenn das Ende der Zeichenkette der Grund ist, dann ist `target[index]` gleich 0, bzw. `'\0'` um das Zeichenäquivalent zu verwenden. In diesem Fall wird `index` auf -1 gesetzt.

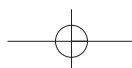
Der Wert von `index` wird an den Aufrufenden zurückgegeben.

Das Ergebnis der Programmausführung sieht wie folgt aus:

```

Der Offset von c in abcdefg ist 2
Der Offset von a in abcdefg ist 0
Der Offset von g in abcdefg ist 6
Der Offset von c in abcdefg ist 2
Der Offset von x in abcdefg ist -1
Press any key to continue

```



390 Anhang B

Das obige Programm kann dadurch vereinfacht werden, dass die Funktion verlassen wird, wenn das Zeichen gefunden wird:

```
int myStrchr(char target[], char testChar)
{
    // Schleife über alle Zeichen der Zeichenkette;
    // spätestens am Ende der Zeichenkette stoppen
    int index = 0;
    while(target[index])
    {
        // wenn das aktuelle Zeichen der
        // Zeichenkette gleich dem gesuchten
        // Zeichen ist ...
        if (target[index] == testChar)
        {
            // ... dann gib Index zurück
            return index;
        }

        // gehe zum nächsten Zeichen
        index++;
    }

    // wenn wir die Schleife durchlaufen haben,
    // sind wir am Ende der Zeichenkette angekommen,
    // ohne das Zeichen zu finden
    return -1;
}
```

Wenn das Zeichen gefunden wird, wird es sofort zurückgegeben. Wenn die Kontrolle die Schleife anders verläßt, kann das nur bedeuten, dass das Ende der Zeichenkette gefunden wurde, ohne das Zeichen zu finden.

Ich persönlich bevorzuge diesen Stil. Es gibt aber Organisationen, bei denen mehrere Rückgaben pro Funktion verboten sind.

- 2. Ein double belegt 8 Bytes, ein int belegt 4 Bytes. Es ist möglich, dass C++ 4 von den 8 Bytes verwendet, um den int-Wert 10 zu speichern, ohne die anderen 4 Bytes zu verwenden. Das verursacht keinen Fehler; sie sollten jedoch nicht davon ausgehen, dass sich Ihr Compiler so verhält.**

- 3. void displayString(char* pszString)**

```
{
    while(*pszString)
    {
        cout << *pszString;
        pszString++;
    }
}
```

4. Die Ausgabe von `pmc->n1` und `pmc->n2` sind vollkommen falsch, weil der Zeiger `pmc` nicht initialisiert wurde, auf etwas zu zeigen. In der Tat kann das Programm abstürzen, ohne überhaupt eine Ausgabe zu erzeugen, wegen dieses nicht initialisierten Zeigers.

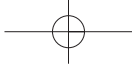
1.2.3 Sonntagmorgen

```
1. class Vehicle
{
    public:
        Vehicle(int nWheels)
        {
        }
};

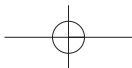
class Motor
{
    public:
        Motor(int nCylinders)
        {
        }
};

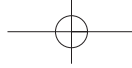
class Car : public Vehicle
{
    public:
        Car(int nCylinders, int nWheels)
            : Vehicle(nWheels), motor(nCylinders)
        {
        }

        Motor motor;
};
```



vakat





Was ist auf der CD-ROM



Die CD-ROM, die diesem Buch beiliegt, enthält Material, das Ihnen bei dem Durcharbeiten der Sitzungen und beim Erlernen von C++ innerhalb eines Wochenendes hilft:

- Installationsdateien für den GNU C++-Compiler
- Alle Programme in diesem Buch

1.1 GNU C++

Die Installationsdateien für das GNU C++, die Sie auf der beiliegenden CD-ROM finden, stammen von der Delorie Website, die in Sitzung 3 erwähnt wurde. Wir stellen die kompletten Dateien für Windows 95, 98, und NT/2000 bereit.

Um GNU C++ mit diesen Dateien zu installieren, führen Sie die folgenden Schritte durch:

1. Erzeugen Sie ein Verzeichnis \DJGPP
2. Kopieren Sie alle Zip-Dateien in dem Ordner auf der CD-ROM, der zu ihrem Betriebssystem passt, in das Verzeichnis DJGPP.
3. Entpacken Sie die Zip-Dateien.
4. Fügen Sie die folgenden Kommandos in Ihre Datei AUTOEXEC.BAT ein:

```
set PATH=C:\DJGPP\BIN;%PATH%
set DJGPP=C:\DJGPP\DJGPP.ENV
```



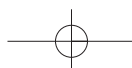
Es wurde angenommen, dass das Verzeichnis DJGPP direkt unter C:\ steht. Wenn Sie Ihr Verzeichnis DJGPP an einer anderen Stelle platziert haben, müssen Sie den Pfad in den obigen Kommandos entsprechend anpassen.

5. Starten Sie Ihr System neu, um die Installation abzuschließen.

Das Verzeichnis \BIN, das beim Entpacken der Dateien erzeugt wurde, enthält die eigentlichen Programme der GNU-Umgebung. Die Datei DJGPP.ENV setzt eine Reihe von Optionen, um die GNU C++-Umgebung zu beschreiben.



Bevor Sie GNU C++ verwenden, stellen Sie sicher, dass in DJGPP.ENV lange Dateinamen angeschaltet sind. Diese Option ausgeschaltet zu haben ist der häufigste Fehler bei der Installation von GNU C++.



394 **Anhang C**

Öffnen Sie die Datei DJGPP.ENV mit einem Texteditor, z.B. mit Microsoft WordPad. Erschrecken Sie nicht, wenn Sie nur eine lange Zeichenkette sehen, die von kleinen schwarzen Kästchen unterbrochen ist. Unix verwendet ein anderes Zeichen für Zeilenumbrüche als Windows. Suchen Sie nach der Phrase »LFN=y« oder »LFN=Y« (Groß- und Kleinschreibung spielt also keine Rolle). Wenn Sie stattdessen »LFN=n« finden (oder »LFN« überhaupt nicht vorkommt), ändern Sie das »n« in »y«. Speichern Sie die Datei. (Stellen Sie sicher, dass Sie die Datei als Textdatei speichern und nicht in einem anderen Format, z.B. als Word .DOC-Datei.)

1.2 Beispielprogramme

Das Verzeichnis \Programs enthält den Sourcecode aller Programme in diesem Buch. Ich empfehle Ihnen, den gesamten Ordner und alle Unterverzeichnisse auf Ihre Festplatte zu kopieren; sie können aber auch einzelne Dateien kopieren, wenn Sie das möchten.

Die Programme sind nach Sitzungen aufgeteilt. Der Ordner einer Sitzung enthält die .CPP-Dateien und die ausführbaren Programme; letztere sollen es dem Leser bequemer machen. Sie können die Programme direkt von der CD-ROM aus ausführen. Die ausführbaren .EXE-Dateien wurden mit GNU C++ erzeugt. Die mit Visual C++ erzeugten .EXE-Dateien befinden sich in einem Unterverzeichnis DEBUG.

Sitzung 2 enthält Anweisungen, wie Programme mit Visual C++ erzeugt werden. Sitzung 3 erklärt das Gleiche für GNU C++.

Hinweis: Zwei Quelldateien wurden abweichend von diesem Buch modifiziert:

1. Die Funktion `ltoa()` wurde durch einen Aufruf der funktional äquivalenten Funktion `itoa()` in `ToStreamWStream.cpp` in Sitzung 29 ersetzt.
2. GNU C++ verwendet den vollständigen Namen von `strstream.h`, während Visual C++ das 8.3-Format `strstrea.h` verwendet. Die Referenz auf diese Include-Datei muss in `ToStreamWStream.cpp` entsprechend angepasst werden. Der Sourcecode verwendet ein `#ifdef`, um zu entscheiden, welche Include-Datei eingebunden werden soll. Der Sourcecode in diesem Buch weist mit einem Kommentar auf dieses Problem hin.

Index

#define Direktive 299 - 302
 #defines __FILE__ 355
 #defines __LINE__ 355
 #else Zweig 302
 #if Anweisung 302
 #if Direktive 302, 303
 #ifdef Direktive 303
 #ifdef Direktive, Debug-Code 304, 305
 #ifdef Direktive, Einschlusskontrolle 303, 304
 #ifndef Direktive, Einschlusskontrolle 303, 304
 #include <iostream.h> Direktive 295
 #include <stdio.h> Direktive 295
 #include 16, 17
 #include Direktive 200, 290, 291, 298, 299
 #include Kommando, Namen, Konstruktion 298, 299
 # (Doppelkreuz) 298, 299
 ! (einfacher logischer Operator) 60
 != (einfacher logischer Operator) 60
 ' _ ' (einfache Anführungszeichen und Unterstrich) 34
 ' ' (einfache Hochkommata) 113
 " " (doppelte Hochkommata) 113, 298, 299
 % (Prozentzeichen) 117
 % mathematischer Operator 53
 %= mathematischer Operator 53
 & (AND bitweiser Operator) 64
 & (unärer) mathematischer Operator 53
 & (unärer) Operator 129
 & (Und-Zeichen) 125
 && (einfacher logischer Operator) 60
 () (Klammern) 81, 82, 157
 , (Komma) 87
 ; (Semikolon) 12, 24, 32
 \ (Backslash) 49
 *= mathematischer Operator 53
 * mathematischer Operator 53
 * (unärer) 53
 */ 31
 / mathematischer Operator 53
 // (doppelter Slash) 31
 ^ (XOR bitweiser Operator) 64
 { } (geschweifte Klammern) 70
 ~ (NOT bitweiser Operator) 65
 ~ (Tilde) 213
 + (unärer) mathematischer Operator 53
 + mathematischer Operator 53
 ++ (unärer) mathematischer Operator 53
 += mathematischer Operator 53
 - (unärer) mathematischer Operator 53
 — (unärer) mathematischer Operator 53
 - mathematischer Operator 53
 -= mathematischer Operator 53
 \ BIN Verzeichnis 19
 FILE 355
 LINE 355
 | (OR bitweiser Operator) 64
 | (Umleitungssymbol) 153

396 **Index**

|| (einfacher logischer Operator) 60
 < (einfacher logischer Operator) 60
 < (Umleitungssymbol) 153
 <= (einfacher logischer Operator) 60
 <> (spitze Klammern) 295, 298, 299
 = (mathematischer Operator) 53
 == (einfacher logischer Operator) 60
 > (einfacher logischer Operator) 60
 > (Umleitungssymbol) 153
 >= (einfacher logischer Operator) 60
 0xff Exit-Code rhide 96

A

Abschneiden von Integer 44, 46
 abstrakte Klassen 267
 Basisklassen 268
 deklarieren 267 - 269
 Klasse konkret machen 269 - 271
 konkrete Elementfunktion 269
 rein virtuelle Funktion 272 - 274
 Syntax rein virtuelle Funktion 267, 268
 Unterklassen 271 - 273
 abstrakte Objekte an Funktionen übergeben 272, 273
 Abstraktion, objektorientierte Programmierung 187, 188
 Absturz, Programme, Funktion concatString() 173
 Abwickeln des Stacks (Ausnahmebehandlung), Code 352, 353
 Account Klasse 266
 konkrete Klasse, aus abstrakter Klasse erzeugen 269, 271
 Syntax rein virtuelle Funktion 267, 268
 Account Programm-Code 267 - 270
 acht-Zeichen-Integer 342
 Add Watch Kommando 169
 addHead() Funktion 217, 218
 Additions-Operatoren 53
 addTail() Funktion 162, 163, 167

Adressen
 nInt, speichern in plnt 131
 Speicher 128, 129
 aktive Klassen 192
 Bereichsauflösungsoperator 196
 Dateien einbinden 199, 200
 definieren 195, 197
 Definitionen definieren 198
 Deklarationen definieren 198
 Include-Dateien 199, 200
 aktive Klassen, Funktionen
 Elemente aufrufen 200 - 203
 Elemente definieren 197, 198
 Elemente überladen 203 - 205
 inline 197, 198
 Namen für Elemente 196, 197
 Nichtelement 195
 aktuelles Verzeichnis 295
 Algorithmen
 definieren 4
 Division-vor-Summe 44
 menschliches Programm 4
 Prozessoren 4
 Alt+F5 Tastaturkürzel 26
 AmbiguousBinding Programm-Code 254, 255
 AmbiguousInheritance Programm-Code 280 - 282
 AND bitweiser Operator (&) 63, 64
 AND Operator (&&) 61, 62
 Anführungszeichen
 " " (doppelte Anführungszeichen) 133, 298, 299
 ' ' (einfache Anführungszeichen) 113
 '_ ' (einfache Anführungszeichen und Unterstrich) 34
 Ansehen von Variablen 175 - 180
 Anweisungen
 #if 302
 ; (Semicolon) 32
 Anzahl von, verglichen mit Anzahl
 Instruktionen 147
 ausführbare, definiert 172
 Ausgabe 92 - 95
 break, switch Kommando 79
 C++-Programme 32
 definiert 32
 Eingabe/Ausgabe 33

- if, demonstriert 69 - 71
- include 31
- inp> Extraktor 337, 338
- Leerraum 32
- return, void-Funktion 85
- Schleifen 71
- switch 79
- Verzweigung 69 - 71
- WRITE 92, 93
- Anwendung, verkettete Liste, Debuggen 176
- Anwendungs-Code aufteilen 291 - 293
- Arbeitsbereich
 - anlegen 11
 - Conversion.pdw Arbeitsbereich 11
 - Kommando (View Menü) 294
 - schließen, Kommando (Menü Datei) 10
- Argumente
 - , (Komma) 87
 - an Funktionen als Zeigervariablen übergeben 133 - 135
 - binäre Operatoren 55, 56
 - binäre Operatoren, Werte verändern 318
 - Default-Werte 223
 - durch Projekteinstellungen übergeben 154
 - Funktionen 84 - 87
 - in Konstruktoren definieren 221, 222
 - in rhide 154
 - Konstruktoren 220 - 223
 - main()-Funktion 152 - 154
 - out, ostream-Objekt 341
 - Programme 150 - 154
 - unärer Operator 55 - 58
 - unärer Operator++() 317, 318
- arithmetische Operatoren 53, 54
- Arrays
 - ArrayDemo Programm-Code 108 - 110
 - Ausgabe-Funktionen 117
 - CharDisplay Programm-Code 111, 112
 - Datenstruktur 159, 160
 - definiert 106
 - Deklarationen 106
 - DisplayString-Programm 112, 113
 - Manipulation von Zeichenketten und Zeiger 145, 146
 - Matrix 111, 112
 - nArray zugreifen 108
 - nInputValue 110
 - parallele 120 - 122
 - wide characters 117
 - zählen 106 - 108
 - Zeichen 111 - 113, 145 - 147
 - Zeichenketten 113, 151, 152
 - Zeichenketten, manipulieren 114 - 117
 - Zeiger 140, 150
 - Zeiger, Differenz zwischen 148 - 150
 - Zeiger-Offsets 142
- Arrays zählen 106 - 108
- auflösen (nicht eindeutig) 204
- aufrufen
 - Elementfunktionen 200 - 203
 - sumSequence()-Funktion 83
- aufteilen
 - Anwendungs-Code 291 - 293
 - Programm 288 - 290
- Ausdrücke
 - C++ Programme 33, 34
 - definiert 34
 - gemischter Modus 50
 - mathematische Operatoren 54
- ausführbare Anweisung, definiert 172
- ausführen
 - Conversion.exe 14, 15
 - GNU C++ Programme 26
 - Programme 14, 15
- Ausgabe
 - Anweisungen 92 - 95, 209
 - C++-Programme 16
 - ConstructElements Programm mit Destruktor 213
 - Funktionen 117
 - Insertter 343, 344
 - Konstruktoren 225
- Ausgabefenster 10
- Ausnahme-Klasse, InvalidArgumentOutOfRangeException::display() 356
- Ausnahmen 350 - 352
- Ausnahmen behandeln
 - Ausnahme-basiere Stream-Ausgabe 335
 - Ausnahme-Klasse
 - InvalidArgumentOutOfRangeException::display() 356
 - catch-Phrasen 354 - 357
 - Fehler behandeln 348, 349

398 **Index**

Fehlerrückgaben 349, 350
 fn(...) Deklaration 354
 Funktionen, fn(...) Deklaration 354
 InvalidArgumentException::display(), Ausnahme-Klasse 356
 Objekte auslösen 354 - 356
 Stack abwickeln, Code 352, 353
 try Schlüsselwort 351
 auto Variable 90

B

\ Backslash () 49
 Basisklassen
 abstrakte Klassen 268
 Destruktor 248
 konstruieren 248
 Bedingungen überprüfen 71
 Begrenzungen
 Gleitkomma-Variablen 46
 int Variablentyp 42 - 45
 Beispiel-Code, Funktionen 81, 83
 Benutzer-Interfaces rhide 20 - 28
 Bereich, begrenzt für int Variable 44
 Bereichsauflösungsoperator 196
 Bibliotheken, MSDN Bibliothek 16
 BIN Verzeichnis 19
 binäre Operatoren 55, 56, 318
 binäre Zahlen 62, 63
 Binden zur Compile-Zeit 253
 Bindung
 Bits, definiert 62
 Compile-Zeit 253
 EarlyBinding, Programm-Code 253
 frühe 253
 späte 255
 bitweise logische Operatoren 58, 63, 64 - 67
 Code zum Testen 65, 66
 Einzelbit-Operatoren 63
 Masken 67
 Sinn 64, 66, 67

BranchDemo Programm-Code 69, 70, 71
 break Anweisung, switch Kommando 79
 break Kommando 76
 BreakDemo Programm-Code 76, 77
 breakpoint Kommando 175
 Build Kommando 169
 Buttons, Neue Textdatei, Textdateien erzeugen 10

C

C++
 Code eingeben 20 - 23
 Compiler 11, 298, 299
 Präprozessor
 C++ Programmiersprache
 Conversion.cpp Programm-Code 30
 EXE-Programme 10
 Fehlermeldungen 14, 25, 26
 GNU C++ 8
 Programm, Ausgabe 16
 Programme, ausführen 14
 Programme, erzeugen 10, 14
 Unterscheidung Groß- und Kleinschreibung 9
 Visual C++ 8
 Zeilen einrücken 9
 CashAccount
 Klasse, abstrakte Unterklasse 271, 272
 Programm-Code 271, 272
 Cast 87
 Cast-Operator 321 - 323
 catch-Phrasen 354 - 357
 CD-ROM
 C++-Wochenend-Crashkurs 8
 Concatenate(Error).cpp Datei 171
 ConstructMembers Programm 213
 Conversion.cpp Datei 10, 21
 factorial() Funktion 348
 FactorialThrow.cpp 355; 365
 StudentID Programm 227
 ToStringWStreams Programm 340; 341
 cerr Objekt 333
 char Variable 47

- CharDisplay Programm-Code 111, 112
 - Checking Klasse 263 - 266
 - cin Eingabe-Objekt 332
 - cin Objekt 333
 - class Schlüsselwort 123
 - class StudentID Elementobjekte, Student Konstruktor 223 - 225
 - ClassData Programm-Code 124, 125
 - clog Objekt 333
 - Code
 - Abwickeln des Stacks (Ausnahme-Behandlung) 352; 353
 - Anwendungen, aufteilen 291 - 293
 - Beispiel-Code 81, 82, 83
 - bitweiser Operator, testen 65, 66
 - C++, Eingabe 20 - 23
 - Cast-Operator überladen 321 - 323
 - cpp Quellcode-Datei 12
 - debuggen, #ifdef Direktive 304, 305
 - display()-Funktion mit Manipulatoren 341
 - displayArray()-Funktion, Integer anzeigen 144, 145
 - factorial()-Funktion 349
 - Konstanten definieren 299, 300
 - leere Dateien erzeugen 21, 22
 - leere Textdateien erzeugen 10, 11
 - Makros definieren 300 - 302
 - Makros Fehler 300 - 302
 - MYNAME Datei öffnen und schreiben 334, 335
 - Objekte auslösen 354 - 356
 - Operatoren als Elementfunktion implementieren 319, 320
 - 0xff Exit-Code rhide 96
 - Student Programm 246 - 248, 290
 - Student-Klasse Elementfunktions, deklarieren außerhalb der Klasse 296
 - USDollar::outpout(), Klasse ostrstream 340, 341
 - Zahlen mitteln 43
 - Zuweisungsoperator überladen 327 - 329
 - Compile-Menü-Kommandos, Make 295
 - Compiler
 - C++ 8, 11, 298, 299
 - Objekte, konvertieren 323
 - Visual C++ 8
 - Computer
 - Concatenate (Error).cpp Datei 171
 - Maschinensprache 23
 - Prozessoren 4
 - Concatenate Programm-Code 114, 115, 171
 - ConcatenatePtr Programm-Code 146, 147
 - concatString()-Funktion
 - Programmabsturz 173
 - Segmentverletzung 175
 - 178, 179
 - ConstructElements-Programm
 - Ausgabe nach Destruktor eingefügt 213
 - CD-ROM 213
 - Code 211, 212
 - Container 168
 - Conventional Klasse 266
 - Conversion Programme, mathematische Operatoren 53
 - Conversion.cpp
 - Datei CD-ROM 10, 21
 - erzeugen 11
 - erzeugen in GNU C++ 24
 - Programm 11
 - Programm-Code 30
 - Conversion.exe
 - ausführen 15
 - Programm 15
 - Conversion.pdw Arbeitsbereich 11
 - CopyStudent Programm-Code 232, 233
 - cout Objekt 332, 333
 - cpp Dateien, mit C++-Präprozessor bearbeiten 298, 299
 - cpp Quellcode-Datei 12
 - Ctrl+F Tastaturkürzel 26
- D**
- Datei Menü Kommandos
 - Arbeitsbereich schließen 10
 - Neu 21
 - Speichern als 22, 23

400 *Index*

- Datei MYNAME öffnen, Code zum Öffnen und Schreiben 334, 335
- Dateien
 - Concatenate (Error).cpp 171
 - Conversion.cpp CD-ROM 10, 21
 - Conversion.exe, ausführen 14, 15
 - Conversion.pdw Arbeitsbereich 11
 - cpp mit C++-Präprozessor bearbeiten 298, 299
 - cpp Quellcode 12
 - DJGPP.ENV 20
 - einbinden 199, 200, 291
 - einbinden, Funktionen 298, 299
 - EXE erzeugen 11
 - fstream.h 334
 - .h #include Direktive 298, 299
 - .h 295
 - Include-Dateien #include Direktive 200
 - ios Konstanten zum Öffnen 334
 - iostream.h Prototypen 333
 - leer, Code zu Erzeugen 21, 22
 - leere Textdatei, Code zum Erzeugen 10, 11
 - myClass.h 303, 204
 - MYNAME, Code zum Öffnen und Schreiben 334, 335
 - Projekt erzeugen in GNU C++ 295
 - Projekt erzeugen in Visual C++ 294
 - Quellmodul 89, 90
 - README.1ST 19
 - SeparatedFn.cpp 292
 - SeparatedKlasse.cpp 288, 289, 291, 292
 - strstrea.h, Definition der Unterklassen ostream 337
 - student.cpp Datei 199
 - Tex, erzeugen 9 - 11
- Daten gruppieren 119 - 122
- Datenelemente
 - Klassen, konstruieren 210 - 212
 - Klassen, static 217, 218
 - Objekte, konstruieren 223 - 228
 - static, Syntax zum Deklarieren 217, 218
 - Syntax zum Deklarieren 228
- DEBUG Parameter 305
- Debug Windows Kommando (View Menü) 179, 180
- debuggen 169
 - Ausgabeeweisungen 92 - 95
 - Code, #ifdef Direktive 304, 205
 - concatString()-Funktion 178, 179
 - debug-Funktion 304, 305
 - debug-Kommandos 170
 - debug-Modus 97
 - Debugger-Programm 92, 93
 - Debugger Visual C++ 179, 180
 - Debugger, Aktion anhalten 173
 - Einzelschritte durch Programme 172, 173
 - ErrorProgram korrigieren 99, 100
 - ErrorProgram 93, 94, 95
 - Fehlertypen 92, 93
 - Funktion 304, 305
 - GNU C++ 96, 97
 - Haltepunkt 175
 - Kommandos 170
 - lokale Variablen ansehen 179, 180
 - Modus 97
 - nNums auswerten 98
 - Probleme, reproduzieren 95
 - Program Reset 174
 - Programme in Einzelschritten 172, 173
 - Programme mit Kommando Step Over 172, 173
 - Release-Modus 97
 - rhide-Debugger 172
 - Segmentverletzung 175
 - Step In Kommando 174
 - Testprogramm 171, 172
 - Variablen modifizieren 175 - 179
 - Variablen, Werte 99
 - verkettete Liste, Anwendung 176
 - Visual C++ 95
 - Visual C++, Debugger 179, 180
 - Visual C++, nNums 95, 96
 - Zeichenketten mit null terminieren 177
 - Zeigerfehler, Ausgabeeweisungen 170
 - Zielzeichenkette 177
 - Zugang der Ausgabeeweisungen 92, 93
- Debugger
 - Aktion anhalten 173
 - Konstruktoren, Ausgabeeweisungen einfügen 209
 - Programm 92, 93
 - Visual C++ 179, 180
- dec Manipulatoren 342
- Default
 - Definition, operator=() 325, 326
 - Konstruktoren 222, 223, 209
 - Werte, Argumente 223

- DefaultStudentID Programm-Code 223 - 225
- definieren
 - aktive Klassen 195 - 197
 - Algorithmen 4
 - Anweisungen 32
 - Arrays 105
 - Ausdrücke 34
 - ausführbare Anweisung 172
 - Bits 62
 - CNU C++ 18
 - Container 168
 - Definitionen 198
 - Deklarationen 32, 198
 - Elementfunktionen 197, 198
 - Faktorianen 265
 - Haltepunkte 175
 - ios::out 334
 - Kommentare 31
 - Konstanten, Code 299, 300
 - Konstruktoren mit Argumenten 221, 222
 - Makros Code 300 - 302
 - Manipulatoren 341
 - nicht uneindeutig 204
 - Objekte 120
 - Operationen auf Zeigertypen 140, 141
 - Operatoren 34
 - ostream Klasse 337
 - Polymorphie 255
 - Programmierung 3, 4
 - Seiteneffekt 229
 - stringstream Klasse 337
 - sumSequence() Funktion 83
 - vertrauenswürdige Funktionen 217
 - Zeichenketten 113
 - Ziffern 62
 - Zugriffsfunktionen 216, 217
- Definitionen
 - definieren 198
 - operator=(), Default 325, 326
 - Deklarationen
 - abstrakte Klassen 267, 268, 269
 - Arrays 105
 - C++-Programme 32, 33
 - Datenelemente, Syntax 228
 - Definition 32, 198
 - Elementfunktionen außerhalb der Klasse 296
 - Elementfunktionen identisch 259, 260
 - linkbare Klassen 160
 - nSum verändern 98
 - Objekte initialisieren 206, 207
 - static Datenelements, Syntax 217, 218
 - Variablen 32, 33
 - von Funktionen 89, 90
- Dekrement-Operatoren 56, 57
- Delorie Web-Site 19
- DemoAssign Programm-Code 327 - 329
- Destruktoren
 - aufrufen 231
 - Ausgabe nach Einfügen in Programm
 - ConstructElements 213
 - Basisklassen 248
 - ofstream Klasse 336
 - virtual 261, 262
 - von Objekten 207, 212, 214
- dezimale Zahlen 42 - 46
- Direktiven
 - #define 299 - 302
 - #if 302, 303
 - #ifdef 303
 - #ifndef, Einschlusskontrolle 303, 304
 - #include <iostream.h> 295
 - #include <stdio.h> 295
 - #include 200, 290, 291
 - #include, .h-Dateien 298, 299
- display() Elementfunktion (virtual), schlaue Inserter 344
- display()-Funktion, Code mit Manipulatoren 341
- displayArray()-Funktion 110, 144, 145
- DisplayString-Programm 112, 113
- displayString()-Funktion 113
- Divisions-Operator 53
- Division-vor-Addition Algorithmus 44
- DJGPP.ENV-Datei 20
- do while-Schleife 72
- Doppelkreuz (#) 298, 299
- Doppelslash (//) 31
- doppelte Hochkommate (") 113
- double-Variable 47, 128
- dumpState()-Funktion 304, 305

402 *Index***E**

EarlyBinding Programm-Code 253

Eigenschaften
 aktive, Klassen hinzufügen 193, 194
 Objekte, Syntax für Zugriff 123
 verkettete Listen 164

einen platten Reifen wechseln
 Algorithmus 4
 Programm 4
 Prozessor 4

einfache Funktionen 85

einfache Hochkommata (') 113

einfache Hochkommata und Unterstrich ('_') 34

einfache logische Operatoren 59 - 62,
 != 60
 && 60
 < 60
 <= 60
 == 60
 > 60
 >= 60

Eingabe von C++-Code 20 - 23

Eingabe/Ausgabe Anweisungen 33

Einschlusskontrolle
 #ifdef Direktive 303, 304

Einzelbit-Operatoren 63

Einzelschritte Programme 172 - 175

Elemente von Klassen konstruieren 230, 231

Elemente am Kopf der Liste angefügt 162, 163

ends Konstante, Inserter-Kommando 338

ErrorProgram 92, 93, 94, 99, 100
 Code 92, 93, 94, 99, 100
 rhide Oxff Exit-Code 96

erzeugen
 Arbeitsbereiche 11
 C++-Programme 10, 11
 Conversion.cpp Programm 11
 Conversion.cpp Programm in GNU C++ 23
 EXE-Dateien 11
 geschützter Kopierkonstruktor 234, 235
 GNU C++ Programm 23 - 26
 leere Dateien 21
 leere Dateien, Code 21, 22

Objekte 189, 190, 206 - 213
 Programme 10, 14
 Projektdateien in GNU C++ 295
 Projektdateien in Visual C++ 294
 Textdateien 9, 11

EXE-Dateien erzeugen 11

EXE-Programme 8

Extraktoren
 inp> Extraktor-Anweisung 337, 338
 operator>() 333

F

F1 key 17, 27

factorial Funktion()
 CD-ROM 348
 Code 348
 Fehlerrückgaben 349

FactorialException Programm-Code 350, 351

FactorialThrow.cpp CD-ROM 355, 356

Faktorisieren
 Definition 265
 Vererbung 263 - 266 - 274

Faktorisieren von Klassen 263
 Account 266
 Checking 263 - 266
 Conventional 266
 Savings 263 - 266

FalseStudentID Programm-Code 225, 226

Feature Autodekrement 72, 73

Fehler
 Ausnahmebehandlung 352, 353
 behandeln 348, 349
 Compile-Zeit 92, 93
 Fehlerrückgaben 348 - 350
 fn(...) Deklaration 354
 Funktionen, fn(...) Deklaration 354
 GNU C++ Installation 24
 Laufzeit 92, 93
 Makros 300 - 302
 Typen von 92, 93
 Zeichenketten, abschließende Null 177
 Zeiger, Ausgabeanweisungen 170

- Fehler, Behandlung
 - Ausnahme-basierte Stream-Ausgabe 335
 - catch-Phrase 354 - 356
 - catch-Phrasen, verbunden mit try-Block 357
 - Objekte, auslösen, catch-Phrasen 354 - 356
- Fehler-Flag
 - filebuf::openprot Wert 334
 - filebuf::sh_none Wert 334
 - filebuf::sh_read Wert 334
 - filebuf::sh_write Wert 2334
 - ifstream Objekt 337
 - ios::ate Konstante 334
 - ios::binärer Konstante 334
 - ios::in Konstante 334
 - ios::nocreate Konstante 334
 - ios::noreplace Konstante 334
 - ios::out Konstante 334
 - ios::trunc Konstante 334
 - nicht null 336
- Fehlermeldung durch null geteilt 95
- Fehlermeldungen 12, 97
 - C++ 12, 25
 - Teilen durch null 95
 - undeklariertes Bezeichner 293
 - Visual C++ 12, 95, 96
- Fenster
 - Arbeitsbereich 9
 - Ausgabe 10
 - Auswerten und Modifizieren 176
 - rhide 22
 - Schließen 9
 - Variablen 179, 180
- FIFO (first-in-first-out) 168
- filebuf::openprot Wert 334
- filebuf::sh_none Wert 334
- filebuf::sh_read Wert 334
- filebuf::sh_write Wert 334
- first-in-first-out (FIFO) 168
- flache Kopien und tiefe Kopien 233, 234
- float-Variable, Speicher 128
- Flusskontrolle, Kommandos 69
 - Autdekrement Feature 72, 73
 - break 76
 - break-Anweisungen 79
 - for-Schleife 74, 75, 76
 - geschachtelte Schleifen 78, 79
 - Schleifen 71 - 77
 - Schleifenkontrollen 76 - 77
 - switch-Anweisung 79
 - switch-Kommando 79
 - Verzweigung 69, 70, 71
- fn()-Funktion 90
- ForDemo Programm-Code 74, 75
- Formate für Klassen 122 - 124
- for-Schleife 74 - 76, 98
- FORTTRAN, WRITE-Anweisung 92, 93
- Free Software Foundation GNU C++ 18
- friends von Klassen 315
- frühe Bindung 253
- fstream-Unterklassen 334 - 337
- fstream.h-Datei 334
- FunktionDemo, Programm-Code 81, 82, 83
- Funktionen 81, 89, 90, 197, 198, 204, 305
 - abstrakte Objekte übergeben 272 - 274
 - addHead() 217, 218
 - addTail() 162, 163, 167
 - Argumente 84 - 87
 - Ausgabe 117
 - Beispiel-Code 81 - 83
 - concatString() 173, 175, 178, 179
 - debuggen 304, 305
 - display(), Code mit Manipulatoren 341
 - displayArray() 110, 144, 145
 - displayString() 113
 - dumpState() 304, 305
 - Einzelschritte 173 - 175
 - factorial() 348, 349
 - fn() 90
 - fn(...) Deklaration 354
 - getData() 167
 - Include-Dateien 199, 200, 298, 299
 - int strcmp(source1 source2) 116
 - int strlen(string) 116
 - int strstr 116
 - konkretes Element 269
 - main() 88-90, 115, 152 - 154
 - Namen 88
 - Nichtelement 195, 320, 321
 - Operatoren 312, 315
 - parseString() 337, 338

404 *Index*

- printf() 117
- protected, Zuweisungsoperator überladen 330, 331
- Prototypen 89, 90
- rationalize() 316
- rein virtuell 267, 268, 272 - 274
- remove() 162, 163
- Rückgabebetyp 85
- setw()-Funktion 342
- someFunktion() 88
- square() 85 - 87
- Step In Kommando 174
- sumArray() 110
- sumSequence(), aufrufen oder definieren 83
- überladen 222, 223
- Variablen speichern 90
- vertrauenswürdige Funktionen, definiert 217
- virtual 259 - 261
- void Schlüsselwort 84, 85, 116
- width() 342
- Zeichenketten manipulieren 116, 117
- Zeigervariablen, Argumente übergeben an 133 - 135
- Zugriffsfunktion, definiert 216, 217
- Funktionen, Elemente 195
 - aufrufen 200 - 203
 - außerhalb der Klasse deklarieren 296
 - definieren 197, 198
 - identisch deklarieren 259, 260
 - Namen vergeben 196, 197
 - operator=() 330
 - Operatoren 321
 - schreiben 198 - 200
 - überladen 204, 205
 - überladen in Unterklassen 252
 - überschreiben 252 - 253
 - Zugriff auf Elemente 201 - 203
- G**
 - gebrochene Werte, Integer 42
 - Geltungsbereich von Variablen 135 - 138
 - gemischte Ausdrücke, definiert 50
 - geschachtelte Schleifen 78, 79
 - getData()-Funktion 167
 - gleich 34
 - Gleichheitsoperator (==) 60
 - Gleitkomma
 - Variablen, Begrenzungen 46
 - Zahlen (floats) 45
 - globale Objekte konstruieren 229, 230
 - GNU C++ 8
 - BIN Verzeichnis 19
 - C++-Code eingeben 20 - 23
 - Conversion.cpp, Programm erzeugen 23
 - Dateien, erzeugen 21
 - definiert 18
 - Delorie Web-Site 19
 - DJGPP.ENV Datei 20
 - Fehlermeldungen 96, 97
 - Free Software Foundation 18
 - Hilfe 27, 28
 - Installation vom Web aus 19, 20
 - Installationsfehler 24
 - Maschinensprache 23
 - Meldungen 24
 - Proektdatei erstellen 295
 - Programme, ausführen 26
 - Programme, erstellen 21 - 23
 - Programme, erzeugen 23 - 26
 - Prozess des Fehlermeldens 24, 25
 - rhide-Fenster 22
 - rhide-Interface 21, 22
 - Windows-Programme entwickeln 27
 - zip-Dateien 19
 - Go Kommando 169
 - Go Menü Kommandos 14
 - Gold-Stern-Programme 92
 - GraduateStudent-Objekt 248 - 250
 - Größer-als-Operator (>) 60, 61
 - Größer-oder-gleich-Operator (>=) 60
 - grundlegender Rahmen von C++-Programmen 30
 - GSInherit Programm-Code 242 - 245

H

- .h-Dateien 295, 298, 299
 - Haltepunkte 175
 - HAS_A Beziehung, GraduateStudent-Objekt 249, 250
 - Heap
 - Container 168
 - Speicher 135 - 138, 234, 316, 317
 - Zeiger 159
 - hex Manipulatoren 342
 - hexadezimale Zahlen 63, 66
 - Hilfe
 - GNU C++ 27, 28
 - MSDN Bibliothek 16
 - Visual C++ 16, 17
 - Hilfe Menü Kommandos, Index 16, 27
 - hinzufügen
 - Ausgabe-Anweisungen im Konstruktor 209
 - Element ans Ende einer Liste 162, 163
 - Objekt am Kopf einer Liste 162
-
- I**
 - if-Anweisung demonstrieren 69 - 71
 - ifstream
 - Klasse, Dateieingabe 336
 - Objekt, Fehlerflag 337
 - Icons, Neue Textdatei 9
 - implementieren
 - SleeperSofa-Klasse 282, 283, 284
 - Vererbung 242 - 245
 - include-Anweisungen 31
 - Include-Dateien 199, 291
 - #include-Direktive 200
 - Funktionen 298, 299
 - Index-Kommando (Hilfe Menü) 16, 27
 - Initialisierung
 - Objekte 206, 207
 - Zeichenketten 113
 - Zeiger 160
 - Inkrement-Operatoren 56 ,57
 - inline
 - Funktionen 197, 198
 - Versionen von Funktionen die nicht funktionieren 305
 - virtuelle Funktionen 261
 - inp> Extraktor-Anweisung 337, 338
 - Insertion-Kommando, ends Konstante 338
 - Installation
 - GNU C++ vom Web aus 19, 20
 - Visual C++ 8
 - Installationsfehler, GNU C++ 24
 - Instanz einer Klasse 189
 - int strcmp(source1 source2) Funktion 116
 - int strlen(string) Funktion 116
 - int strstr Funktion 116
 - int Variable
 - begrenzte Bereiche 45
 - Begrenzungen 42 - 44
 - Division-vor-Addition Algorithmus 44
 - Integer runden 42 - 45
 - Integer abschneiden 44, 46
 - Speicher 128
 - Integer
 - Abschneiden 44
 - Abschneiden, Problem lösen 44, 46
 - displayArray() Funktion Code 144, 145
 - gebrochene Werte 42
 - Runden 42 - 45
 - Integrität von Objekten 206
 - Interface rhide 21, 27, 28
 - InvalidArgumentOutOfRangeException::display(), Ausnahme-Klasse 356
 - ios-Klasse
 - ios::out, definiert 334
 - Konstanten zum Öffnen von Dateien 334
 - ios::ate, Konstante 334
 - ios::binary, Konstante 334
 - ios::in, Konstante 334
 - ios::nocreate, Konstante 334
 - ios::noreplace, Konstante 334
 - ios::out, definiert 334
 - ios::out, Konstante 334

406 **Index**

ios::trunc, Konstante 334
 iostream.h Datei, Prototypen 333
 istrstream Klasse, definiert 337

K

Kapselung 288, 289
 Klammern () 69, 70, 81, 82, 157
 Klassen 125
 Account 266 - 268
 aktive 192
 aktive, definieren 193 - 197
 Ausnahme, InvalidArgumentException::display()
 356
 Basisdestruktor 248, 268
 Bereichsauflösungsoperator 196
 CashAccount, abstrakte Unterklasse 271, 272
 Checking 263 - 266
 Conventional 266
 Dateien, einbinden 199, 200
 Datenelemente konstruieren 210 - 212, 217,
 218, 223 - 228
 Definitionen, definieren 198
 Deklarationen, definieren 198
 Destruktoren von Objekten 212, 213
 Elemente erzeugen 230, 231
 Faktorisieren 263 - 266
 Formate 122 - 124
 friend 315
 fstream Unterklassen 334 - 337
 ifstream, Dateieingabe 336
 Include-Dateien 199, 200
 Inline-Funktionen 197, 198
 Instanzen 189
 ios, ios::out definiert 334
 istrstream, definieren 337
 konkrete aus abstrakter Klasse erzeugen 269 -
 271
 konkrete Unterklassen, Implementierung rein vir-
 tueller Funktionen 273
 linkbar, definieren 160
 Nichtelement-Funktionen 195
 Objekt, Zugriff auf Eigenschaften 123
 ofstream 334, 336
 ostrstream USDollar::outputput()-Code 340;341
 ostrstream, definieren 337
 Savings 263 - 266
 Savings, abstrakte Unterklasse 271, 272
 selbstenthaltend 190, 191
 Sleepersofa, implementieren 282 - 284
 strstream, definieren 337
 struct Schlüsselwort, Grenzen 193, 194
 Struktur 192, 193
 Student, Konstruktoren 222, 223
 Superklasse 189
 Unterklassen 189, 246, 247 - 249, 253
 vertrauenswürdige Funktionen, definiert 217
 Zugriffsfunktionen, definiert 216, 217
 Zusammenfassung 192 - 194
 Zuweisungsoperator, mit geschützter Funktion
 überladen 330, 331
 Klassen aktive Eigenschaften hinzufügen 193, 194
 Klassen, abstrakt 271, 272
 Basis 268
 deklarieren 267 - 269
 konkrete Elementfunktion 269
 rein virtuelle Funktionen 267, 268, 273, 274
 rein virtuelle Funktionen überladen 272
 Klassen, Elementfunktionen 195
 aufrufen 200 - 203
 definieren 197, 198
 deklarieren außerhalb von 296
 Namen vergeben 196, 197
 schreiben 198 - 200
 überladen 204, 205
 Klassen, Funktionen
 Element aufrufen 200 - 203
 Element definieren 197, 198
 Element 195
 inline 197, 198
 Element Namen geben 196, 197
 Nichtelement 195
 Element überladen 204, 205
 Klassen, Vererbung
 implementieren 242 - 245
 konstruieren Unterklassen 246 - 249
 public Schlüsselwort 241
 Vorteile 241
 Klassifizierung in der objektorientierten
 Programmierung 188, 189
 Kleiner-als-Operator (<) 60

- Kleiner-oder-gleich-Operator (<=) 60
- Komma (,) 87
- Kommandos
 - #include Namen konstruieren 298, 299
 - Add Watch 169
 - break 76
 - Build 169
 - C++-Programme 31
 - debuggen 170
 - definiert 32
 - Flusskontrolle 69
 - Go 169
 - Go Menü 14
 - Haltepunkt 175
 - Haltepunkt setzen 169
 - Insertter, Konstante ends 338
 - Program Reset 169
 - Schleife, innere Schleife 78, 79
 - Step In 169, 174
 - Step Over 169
 - Step Over, Programme debuggen 172, 173
 - switch()-Kontrolle 151
 - switch, break-Anweisungen 79
 - View User Screen 169
 - View Variable 169
- konkrete Element-Funktion 269
- konkrete Klasse aus abstrakter Klasse erstellen 269 - 271
- konkrete Unterklassen, rein virtuelle Funktionen implementieren 273
- Konstanten
 - #define Direktive 299, 300
 - Code zum Definieren 299, 300
 - ends, Insertter-Kommando 338
 - ios::ate 334
 - ios::binärer 334
 - ios::in 334
 - ios::nocreate 334
 - ios::noreplace 334
 - ios::out 334
 - ios::trunc 334
 - Variablen 48
- konstruieren
 - Basisklassen 248
 - Datenelemente von Klassen 210 - 212
 - Datenelement-Objekte 223 - 228
 - Elementobjekte mit Klassentyp 223 - 228
 - globale Objekte 229, 230
 - Klassenelemente 230, 231
 - lokale Objekte 229
 - Namen in #include Kommando 298, 299
 - Objekte mehrfacher Vererbung 285
 - static Objekte 229
 - Unterklassen 246 - 249
- Konstruktoren
 - Argumente 220 - 223
 - Ausgabe 225
 - Ausgabeansweisungen einfügen 209
 - Begrenzungen 209, 210
 - Datenelemente, Syntax der Deklaration 228
 - Default 209, 222, 223
 - Destruktoren, aufrufen 231
 - flache Kopie gegen tiefe Kopie 233, 234
 - globale Objekte konstruieren 229, 230
 - Heap-Speicher 234
 - Kopier- (protected) erzeugen 234, 235
 - Kopier- 231, 232
 - lokale Objekte, konstruieren 229
 - ofstream Klasse 334
 - Reihenfolge der Konstruktion 228 - 231
 - Seiteneffekt, definiert 229
 - static Objekte, konstruieren 229
 - Student-Klasse 222, 223
 - Student-Klasse, StudentID Element-Objekte 223 - 225
 - überladen 221, 222
 - Vergleich Kopieren und Zuweisen 326, 327
 - virtual 261
 - void 209, 222, 223
 - von Objekten 207
- Kontrolle für Schleifen 75 - 77
- Konvertieren von Objekten 323
- Kopfzeiger auf Liste, Objekte einfügen 161
- Kopierkonstruktor 231, 232
 - Kopie, flache gegen tiefe 233, 234
 - protected, erzeugen 234, 235
 - Vergleich mit Zuweisungsoperator 326, 327
- kurze Schaltkreise 62
- Kurznamen von Funktionen 88

408 **Index****L**

last-in-first-out (LIFO) 168
 LateBinding-Programm-Code 257 - 259
 Laufzeitfehler 92, 93
 Layout Programm-Code 128, 129
 LayoutError Programm-Code 132, 133
 leere Datei, Code zur Erzeugung 21, 22
 leere Textdatei, Code zur Erzeugung 10, 11
 Leerraum (Anweisungs-) 32
 Lesbarkeit, verbessern mit Überladen von Operatoren 311, 312
 LFN=y Phrase 21
 LIFO (last-in-first-out) 168
 linkbare Klassen deklarieren 160
 LinkedListData Programm-Code 165 - 167
 Links-Shift (<<) Operator 311, 312
 Links-Shift Operator 346, 347
 logische Operatoren
 ! (einfacher) 60
 != (einfacher) 60
 && (einfacher) 60
 & (einfacher) 60
 < (einfacher) 60
 <= (einfacher) 60
 == (einfacher) 60
 > (einfacher) 60
 >= (einfacher) 60
 bitweise 58, 63 - 67
 bitweise, binäre Zahlen 63
 einfacher 58, 60 - 62
 kurze Schaltkreise 62
 logische Variablen 62
 lokale Objekte konstruieren 229
 lokale Variablen ansehen 179, 180
 long-Variable 47, 128

M

main()-Funktion 88 - 90, 115, 152, 154
 Make Kommando (Compile Menü) 295

Makros

#define Direktive 299, 300 - 302
 Code zum Definieren 300 - 302
 Fehler 300 - 302

Manipulation von Zeichenketten: Arrays gegen Zeiger 145, 146

Manipulatoren

dec 342
 definieren 341
 display()-Funktion 341
 hex 342
 oct 342
 setfill(c) 342
 setprecision(c) 342
 setw(n) 342
 stream I/O 341 - 343

Manipulieren von Zeichenketten 114 - 117

Maschine

Anzahl Instruktionen ind Anzahl Anweisungen 147
 Sprachen 23

Masken 67

mathematische Operatoren 53

% 53
 %= 53
 - (unär) 53
 — (unär) 53
 -, 53
 * 53
 *= 53
 /, 53
 + (unär) 53
 ++ (unär) 53
 += 53
 = 53
 -= 53
 arithmetische 53, 54
 Ausdrücke 54
 binärer 55, 56
 Dekrement 56, 57
 Inkrement 56, 57
 unär 55 - 58
 Zuweisung 57, 58

mathematischer Operator 53

Matrix, Arrays 111, 112
 aufrufen 200- 203

- definieren 197, 198
 - deklarieren, außerhalb der Klasse 296
 - deklarieren, identisch 259, 260
 - display() (virtual), schlaue Inserter 344
 - Elemente zugreifen 201 - 203
 - Namen geben 196, 197
 - nicht uneindeutig, definiert 204
 - operator=() 330
 - Operatoren 321
 - schreiben 198 - 200
 - Student-Klasse, Code außerhalb der Klasse deklarieren 296
 - überladen 204, 205
 - überladen in Unterklassen 252
 - überschreiben 253
 - mehrfache Vererbung
 - Nachteile 285, 286
 - Objekte konstruieren 285
 - SleeperSofa-Programm 277, 278
 - SleeperSofa-Speicheranordnung 281, 282, 284
 - Uneindeutigkeit 278, 279
 - virtuelle Vererbung 279 - 285
 - mehrfache Vererbung 276
 - Meldung von Fehlern
 - C++ 25
 - Prozess des Meldens 24, 25
 - menschliches Programm
 - Algorithmus 4
 - Programme 4
 - Prozessoren 4
 - Mischen von Variablen 50, 51
 - Mittlung Programm, mathematische Operatoren 53
 - mittleren Grad berechnen (dAverage) 193, 194
 - Modelle (Paradigmas) 191
 - Modi, Debug oder Release 97 97
 - Module 89, 90, 288, 289
 - Module linken 288, 289
 - modulo-Operator 53, 54
 - MSDN Bibliothek 16
 - MS-DOS Backslash () 303, 304
 - MultipleVirtual Programm-Code 282 - 284
 - Multiplikation, Operatoren 53
 - MyClass, myclass.h Datei 303, 304
 - MYNAME-Datei, Code zum Öffnen und Schreiben 334, 335
 - MySpecialClass-Objekt, Debug-Funktion 304, 305
- ## N
- Name& (Rückgabety) operator=() 329, 330
 - NamedStudent Programm-Code 221, 222
 - Namen
 - konstruieren in #include Kommando 298, 299
 - von Funktionen 88
 - Namen vergeben
 - Elementfunktionen 196, 197
 - Variablen 33, 51
 - Zeichenketten 113
 - nArray, Zugriff 108
 - NestedDemo Programm-Code 78, 79
 - Neu Kommando (Datei Menü) 21
 - Neue TextDatei
 - Button, neue Textdatei erzeugen 10
 - Icon 9
 - nGlobal Variable 90
 - nicht initialisierte verkettete Zeiger 167
 - nicht null Fehler-Flag 336
 - nicht uneindeutig, definiert 204
 - Nichtelement-Funktionen 195, 320, 321
 - nInputValues Array 110
 - nInt Adresse, speichern in plnt 131
 - nLocal Variable 90
 - nLoopCount Variable 72
 - nNums 95, 96, 98
 - NOT bitweiser Operator (~) 64
 - NOT Operator (!) 60
 - nStatic Variable 90
 - nSum
 - auswerten 98
 - deklarieren, ändern 98
 - null (terminieren), Zeichenketten 177
 - numerische Typen, Cast 87
 - nValue, auswerten 98

410 *Index***O**

Objekte

- abstrakt, an Funktionen übergeben 272, 273
- am Kopfzeiger der Liste eingefügt 162
- auslösen `__FILE__` 355
- auslösen `__LINE__` 355
- auslösen, catch Phrasen 354 - 356
- cerr 333
- cin 333
- cin Eingabe 332
- clog 333
- cout 332, 333
- Datenlemente 210 - 212, 225, 226
- definiert 120
- Destruktoren 207, 212, 213
- Eigenschaften, Syntax für Zugriff 123
- erzeugen 189, 190, 206 - 212, 214
- globale, konstruieren 229, 230
- GraduateStudent 248 - 250
- identifizieren in objektorientierter Programmierung 188, 189
- ifstream, Fehler-Flag 337
- initialisieren 206, 207
- Integrität 206
- Klasse StudentID Element, Student Konstruktor 223 - 225
- Konstruktor 207
- Konstrukturen, Begrenzungen 209, 210
- konvertieren 325
- Kopierkonstruktor im Vergleich mit Zuweisungsoperator 326, 327
- lokale, konstruieren 229
- Manipulatoren 341
- Mehrfachvererbung, konstruieren 285
- MySpecialClass-Objekt, Debug-Funktion 304, 305
- ostream, out Argument 341
- out 337, 338
- parallele Arrays 120 - 122
- Reihenfolge der Konstruktion 228 - 231
- static, konstruieren 229, 230
- Streaml/O 333
- temporäre 317
- übergeben 157 - 159
- vernichten 206 - 212, 214
- vertrauenswürdige Funktionen, definiert 217
- Zeiger 156 - 159
- Zugriffsfunktion, definiert 216, 217
- Zugriffskontrolle 214, 217, 218
- Zugriffskontrolle, private Schlüsselwort 216
- Zugriffskontrolle, protected Schlüsselwort 214 - 217
- Zugriffskontrolle, public Schlüsselwort 214
- Zuweisungsoperator, Vergleich mit Kopierkonstruktor 326, 327
- Objekte auslösen
 - `__FILE__` 355
 - `__LINE__` 355
 - catch Phrasen 354 - 356
- Objekte identifizieren, in objektorientierter Programmierung 188, 189
- Objekte übergeben 157 - 159
- objektorientierte Programmierung
 - Abstraktion 187, 188
 - Abstraktionslevel 188
 - Kapselung 288, 289
 - Klassen, selbstenthaltend 190, 191
 - Klassifizierung 188, 189
 - Objekte, erzeugen 189, 190
 - Objekte, identifizieren 188, 189
 - Paradigmas 191
 - Polymorphie 255 - 257
- oct Manipulatoren 342
- Offsets, Zeiger-Arrays 142
- ofstream Klasse
 - Destruktor 336
 - Konstruktor 333
- ofstream, Konstruktor 335
- Operationen
 - definiert auf Zeigertypen 140, 141
 - Zeiger + Offset 140, 141
 - Zeiger-Offset 140, 141
 - Zeiger2 Zeiger1 140, 141
 - Zeigertypen 148
- Operator, Operatoren überladen
 - Beziehungen dazwischen 316
 - binäre, Verändern von Argumentwerten 318
 - Cast 321 - 323
 - Code als Elementfunktion 319, 320
 - Elementfunktionen 321
 - Funktionen, Beziehungen dazwischen 312
 - Nichtelement-Funktionen 321
 - unärer operator++(), Argumente 317, 318

- operator+() 315 - 318
 - operator++()-Funktion 315
 - operator<<() 333, 343, 344
 - Ausgabe 343, 344
 - schlau 344 - 347
 - USDollar Programm-Code 343, 344
 - operator<<() (Insertor), Stream I/O 333
 - operator=()
 - Default-Definition 325, 326
 - Elementfunktion 330
 - Name& (Rückgabotyp) 329, 330
 - Operator>() (Extraktor) Stream I/O 332
 - Operatoren
 - Addition 53
 - AND (&&) 60 - 61
 - AND 63
 - arithmetische 53, 54
 - Bereichsauflösung 196
 - Beziehung dazwischen 316
 - binäre 55, 56, 318
 - Cast 321 - 323
 - Code als Elementfunktion 319, 320
 - definiert 34
 - Dekrement 56, 57
 - Division 53
 - einfacher logischer 73, 62
 - Einzelbit 63
 - Elementfunktionen 321
 - Funktionen, Beziehungen dazwischen 312
 - Gleichheit (==) 60
 - größer als (>) 60 - 62
 - größer oder gleich (>=) 60, 61
 - Inkrement 56, 57
 - kleiner als (<) 60, 61
 - kleiner oder gleich (<=) 60, 61
 - Links-Shift 346, 347
 - Links-Shift (<<) 311, 312
 - logische 62 - 67
 - mathematische 53
 - modulo 53, 54
 - Multiplikation 53
 - Nichtelement-Funktions 321
 - NOT (!) 60
 - operator+() 316 - 318
 - operator<< (Insertor) Stream I/O 333
 - operator<<() 343,344
 - operator>() (Extraktor) Stream I/O 333
 - OR () 60, 64
 - Postfix x++ 316
 - Präfix ++x 316
 - Rechts-Shift (>)311, 312
 - Shift 346;347
 - Subtraktion 53
 - überladen, operator<<() 333
 - unäre 55 - 58
 - unärer operator++(), Argumente 317, 318
 - Ungleichheit (!=) 60
 - XOR 64
 - Zeiger 129
 - Zuweisung (=) 311, 312
 - Operatoren bitweise
 - Code zum Testen 65, 66
 - logische 58, 63 - 67
 - Masken 67
 - Zweck 66, 67
 - Operatoren, Zuweisung 34, 57
 - Code zum Überladen 327 - 329
 - überladen 325 - 330
 - Vergleich mit Kopierkonstruktor 326, 327
 - OR bitweiser Operator (|) 64
 - OR Operator (||) 60
 - OR Operator 64
 - ostream-Objekt, out Argument 341
 - ostrstream Klasse
 - defining 337
 - USDollar::output() Code 340, 341
 - out Argument, ostream Objekt 341
 - out Objekt 337, 338
- P**
- p (Präfix) Zeiger Variablen 130
 - Paradigmas 191
 - parallele Arrays 120 - 122
 - ParallelData Programm-Code 120 - 122
 - Parameter
 - DEBUG 305
 - parseString()-Funktion 337, 338
 - PassObjektPtr-Programm-Code 157 - 159

412 *Index*

- Phrase LFN=y 21
- plnt, speichert Adresse von nlnt 131
- platter Reifen
 - Algorithmus zum Wechseln 4
 - Programm zum Wechseln 4
 - Prozessor zum Wechseln 4
 - Polymorphie 252
- abstrakte Objekte, an Funktionen übergeben 272, 273
 - AmbiguousBinding Programm 254, 255
 - definieren 255
 - Destruktoren, virtual 261, 262
 - Elementfunktionen, identisch deklarieren 259, 260
 - Elementfunktionen, überschreiben 252, 253
 - Konstruktoren, virtual 261
 - objektorientierte Programmierung 256, 257
 - virtuelle Schlüsselwort 257
 - virtuelle Funktionen 260, 261
 - virtuelle Funktionen, Zeiger 261
- Postfix-Operator x++ 316
- Präfix-Operator ++x 316
- Präprozessor 298, 299
 - #constants, Code zum Definieren 299, 300
 - #define-Direktive 299 - 302
 - #else-Zweig 302
 - #if-Anweisung 302
 - #if-Direktive 302, 303
 - #ifdef-Direktive 303 - 305
 - #include Direktive 298, 299
 - #include Kommando 298, 299
 - #macros 300 - 302
 - DEBUG-Parameter 305
 - dumpState()-Funktion 304, 305
 - Einschlusskontrolle, #ifdef Direktive 303, 304
 - Funktionen, Inline-Versionen funktionieren nicht 305
 - MyClass, myclass.h-Datei 303, 304
- printf()-Funktion 117
- private Schlüsselwort, Zugriffskontrolle auf Objekte 216
- Probleme, reproduzieren 95
- Program Reset-Kommando 169
- Program Reset, Debug-Operationen 174
- Programme
 - abstürzende concatString()-Funktion 173
 - aktuelles Verzeichnis 295
 - Anweisungen 32
 - Anwendungs-Code, aufteilen 291 - 293
 - Argumente 150 - 154
 - Ausdrücke 33, 34, 54
 - Average, mathematische Operatoren 53
 - binäre Zahlen 62, 63
 - ConstructElements, Ausgabe mit Destruktor 213
 - Conversion.cpp 11, 23
 - Debuggen 169, 171 - 173
 - Deklarationen 32, 33
 - dezimale Zahlen 42 - 46
 - DisplayString 112, 113
 - Division-vor-Addition Algorithmus 44
 - Eingabe/Ausgabe-Anweisungen 33
 - Einzelschritte 172, 173
 - elementarer Programmrahmen 30
 - erzeugen 9, 10, 11, 14, 18
 - EXE 10
 - gemischte Ausdrücke 50
 - gleich 39
 - Gleitkomma-Zahlen 45
 - GNU C++ 18, 20 - 26
 - Gold-Stern 92
 - Groß- und Kleinschreibung unterscheiden 32
 - h-Dateien 295
 - hexadezimale Zahlen 63, 66
 - include-Anweisungen 31
 - Integer 42, 44 - 46
 - Kapselung 288, 289
 - Kommentare definieren 31
 - kurze Schaltkreise 62
 - Leerraum 32
 - Masken 67
 - menschliches Programm 4
 - Module 288, 289
 - MS-DOS Backslash (\) 49
 - platte Reifen wechseln 4
 - Projektdateien 293 - 295
 - Rahmen 32
 - SeparatedClass.cpp-Datei 288, 289, 291, 292
 - Student 246 - 248
 - StudentID CD-ROM 227
 - teilen 288 - 290
 - Uneindeutigkeit vermeiden 34
 - Windows mit GNU C++ entwickeln 27

- Programme, Code
 - Account 267 - 270
 - AmbiguousBinding 254, 255
 - AmbiguousVererbung 280, 281, 282
 - ArrayDemo 108 - 110
 - BranchDemo
 - BreakDemo
 - CashAccount 271;272
 - CharDisplay 111, 112
 - ClassData 124, 125
 - Concatenate 114, 115, 171
 - ConcatenatePtr 146, 147
 - ConstructElements 211, 212
 - Conversion 30
 - CopyStudent 232, 233
 - DefaultStudentID 223 - 225
 - DemoAssign 327 - 329
 - EarlyBinding 253
 - Error Program 92 - 94, 99, 100
 - FactorialException 350, 351
 - FalseStudentID 226
 - ForDemo 74, ForDemo 75
 - FunktionDemo 81 - 83
 - GSInherit 243 - 245
 - LateBinding 257 - 259
 - Layout 128, 129
 - LayoutError 132, 133
 - LinkedListData 165 - 167
 - MultipleVirtual 282 - 284
 - NamedStudent 221, 222
 - NestedDemo 78, 79
 - ParallelData 120 - 122
 - PassObjektPtr 157 - 159
 - ProtectedElements 216
 - SeparatedClass 288 - 290
 - SleeperSofa 277, 278
 - SquareDemo 85 - 87
 - Student 246 - 248, 290
 - Student Klasse 296
 - USDollar 319, 320, 338 - 340
 - USDollarAdd 313 - 315
 - USDollarCast 321 - 323
 - VirtualInserter 344 - 346
 - WhileDemo 71, 72
- Programme, Operatoren 34
 - arithmetische 53, 54
 - binäre 55, 56
 - bitweise 58, 63 - 67
 - Dekrement 56, 57
 - einfache logische 58, 60 - 62
 - Einzelbit 63
 - Inkrement 56, 57
 - logische 58
 - unäre 55 - 58
 - Vorrang 55, 56
 - Zuweisung 34, 57, 58
- Programme, Variablen 33, 34, 41
 - char 47
 - double 47
 - Grenzen von Gleitkomma 46
 - int 42 - 45
 - Konstanten 48
 - logische 62
 - long 47
 - mischen 50, 51
 - Namen geben 33, 51
 - Sonderzeichen 48, 49
 - Typen 46 - 49
 - Zeichenketten 47
- Programmierung
 - Computerprozessoren 4
 - definieren 3, 4
 - menschliches Programm 4
 - Sprachen, C++ 8
- Projektdateien 293
 - erzeugen in GNU C++ 295
 - erzeugen in Visual C++ 294, 295
- Projekteinstellung, Argumente übergeben 154
- protected-Funktionen, überladener Zuweisungsoperator 330, 331
- protected keyword, Zugriffskontrolle für Objekte 214 - 217
- ProtectedElements Programm-Code 216
- Prototypen
 - Funktionen 89, 90
 - iostream.h Datei 333
- Prozentzeichen (%) 117
- Prozessoren
 - Computer 4
 - menschliches Programm 4
 - platten Reifen wechseln 4
 - Programmausgabe 16

414 **Index**

public Schlüsselwort
 Vererbung 245
 Zugriffskontrolle für Objekte 214
 public Schlüsselwort 123, 192, 193

Q

Quellcode-Dateien,
 Quellcode-Dateien, cpp 12
 Quelldateien, Module 89, 90

R

Rahmen für C++-Programme 30, 32
 rationalize() Funktion 316, 317
 README.1ST Datei 19
 Rechts-Shift (>) Operator 311, 312
 Referenzen, Zeiger 158, 159
 Reihenfolge der Konstruktion von Objekten 228 -
 231
 rein virtuelle Funktionen 274
 Features in konkreten Unterklassen implementie-
 ren 273
 Syntax 267, 268
 überladen 272
 Zweck bitweiser Operatoren 66, 67
 Release-Modus 97
 remove()-Funktion 162, 163
 return-Anweisung, void-Funktion 85
 rhide
 Benutzer-Interface, GNU C++ Hilfe 27, 28
 Debugger 172
 Fenster 22
 Interface 21
 Oxff Exit-Code 96
 Programmargumente 154
 Rückgabety, Funktionen 85
 Runden von Integer 42 - 45
 Rundungsfehler 46

S

Savings-Klasse 263, 264
 abstrakte Unterklasse 271, 272
 konkrete Klasse, erzeugen aus abstrakter Klasse
 269, 270
 schlaue Inserter 344 - 347
 Schleifen
 Bedingungen überprüfen 71
 do while 72
 endlos 73, 74
 for 74 - 76
 for modifizieren 98
 geschachtelte 78, 79
 Kontrollen 76, 77
 nLoopCount-Variable 72
 Variablen 77
 while 71 - 74, 115
 Schleifen-Kommandos oder Anweisungen 71 - 77
 Schleifen-Kommandos, geschachtelte Schleifen 78,
 79
 Schlüsselwörter
 Klasse 123
 private, Zugriffskontrolle auf Objekte 216
 protected, Zugriffskontrolle auf Objekte 214,
 215, 217
 public 123, 192, 193
 public, Vererbung 245
 public, Zugriffskontrolle auf Objekte 214
 struct 123, 192, 193
 try 351
 virtual 257, 258, 284
 void 84
 schreiben
 Elementfunktionen 198 - 200
 MYNAME Datei, Code zum Öffnen und Schrei-
 ben 334, 335
 Segmentverletzung 175
 Seiteneffekt 229
 selbstenthaltende Klassen 190, 191
 Semikolon (;) 12, 24, 32
 SeparatedClass Programm-Code 288 - 290
 SeparatedClass.cpp-Datei 288, 289, 291 - 293
 SeparatedFn.cpp-Datei 292
 Set Breakpoint-Kommando 169

- setfill(c)-Manipulatoren 342
- setprecision(c)-Manipulatoren 342
- setw()-Funktion 342
- setw(n)-Manipulatoren 342
- shift-Operatoren 346;347
- Slash
 - \ (Backslash) 49
 - // (doppelter) 31
- SleeperSofa
 - Klasse, Implementierung 282 - 284
 - Programm-Code 277, 278
 - Speicheranordnung 281, 282, 284
- Software, Free Software Foundation GNU C++ 18
- someFunktion() 88
- Sonderzeichen, Variablen 48, 49
- späte Bindung 255
- Speicher
 - Adressen 128, 129
 - double Variable 128
 - float Variable 128
 - Heap 135 - 138, 234, 316, 317
 - int Variable 128
 - long Variable 128
 - SleeperSofa, Speicheranordnung 281, 282, 284
- Speicher als Kommando (Datei Menü) 22, 23
- spitze Klammern < > 295, 298, 299
- Sprachen
 - Maschine 23
 - Programmieren in C++ 8
- square()-Funktion 85 - 87
- SquareDemo Programm-Code 85 - 87
- static-Datenelemente
 - Klassen 217, 218
 - Syntax zur Deklaration 217, 218
- static-Objekte, konstruieren 229
- Step In-Kommando 169, 174
- Step Over-Kommando 169, 172, 173
- Stream I/O
 - Ausgabe Insertter 343, 344
 - cin Eingabe-Objekt 332
 - cout Ausgabe-Objekt 332, 333
 - display()-Elementfunktion (virtual), schlaue Insertter 344
 - display() Funktion, Code mit Manipulatoren 341
 - ends-Konstante, Insertter-Kommando 338
 - fstream Unterklassen 334 - 337
 - fstream.h-Datei 334
 - ifstream-Klasse, Dateieingabe 336
 - inp> Extraktor-Anweisung 337, 338
 - Insertter 343 - 347
 - ios-Klasse, Konstanten zum Öffnen einer Datei 334
 - ios::out, definiert 334
 - iostream.h-Datei 333
 - istream-Klasse, definiert 337
 - Klassen, ofstream, Konstruktoren 333
 - Links-shift-Operator 346; 347
 - Manipulatoren 341 - 343
 - MYNAME-Datei, Code zum Öffnen und Schreiben 334, 335
 - nicht null Fehler-Flag 336
 - Objekte 333
 - ofstream-Klasse, Destruktor 336
 - ofstream-Klasse, Konstruktoren 334 - 335
 - Operator<<() (Insertter) 333
 - Operator<<() 343;344
 - Operator>() (Extraktor) 332
 - ostream Klasse, definieren 337
 - out-Objekt 337, 338
 - parseString()-Funktion 337, 338
 - Prototypen, iostream.h-Datei 333
 - schlaue Insertter 344 - 347
 - setw()-Funktion 342
 - Shift-Operatoren 346, 347
 - Stream-Ausgabe, Ausnahme-basierte Fehlerbehandlung 335
 - strstrea.h-Datei, definieren strstream-Unterklassen 337
 - strstream-Klasse, definiert 337
 - strstream-Unterklassen 337 - 341
 - USDollar::output()-Code 340;341
 - VirtuellInsertter Programm-Code 344 - 346
 - width-Parameter 342
 - width()-Funktion 342
 - Zeichenketten, behandeln 338 - 341
- strstrea.h-Datei, definiert strstream-Unterklassen 337
- strstream
 - Klasse, definieren 337
 - Unterklassen 337 - 341

416 Index

- struct-Schlüsselwort 123, 192 - 194
 - Struktur, Klassen 192, 193
 - Student-Klasse
 - Elementfunktionen, Code zur Deklaration außerhalb der Klasse 296
 - Konstruktoren 222, 223
 - Student-Konstruktor, Klasse StudentID-Element-Objekt 223 - 225
 - Student Programm-Code 246 - 248, 290
 - student.cpp-Datei 199
 - StudentID-Programm CD-ROM 227
 - Subtraktions-Operatoren 53
 - sumArray()-Funktion 110
 - sumSequence()-Funktion, aufrufen oder definieren 83
 - Superklassen 189
 - switch
 - Anweisung 79
 - Kommando, break-Anweisungen 79
 - switch(), Kontrollkommando 151
 - Syntax
 - Datenelement deklarieren 228
 - Objekteigenschaften, zugreifen 123
 - rein virtuelle Funktionen 267, 268
 - statische Datenelemente, deklarieren 217, 218
- T**
- Tastaturkürzel
 - Alt+4 179, 180
 - Alt+F4 26
 - Ctrl+F5 14
 - Ctrl+F9 26
 - Tasten 17, 27
 - temporäre Objekte 317
 - terminieren
 - null, Zeichenketten 177
 - Zeichenketten 115
 - testen
 - bitweise Operatoren, Code 65, 66
 - Programm debuggen 171, 172
 - Textdateien, erzeugen 9 - 11
 - tiefe Kopie gegen flache Kopie 233, 234
 - Tilde (~) 132
 - Ton, Visual C++ 12
 - ToStreamWStreams-Programm CD-ROM 340, 341
 - try-Schlüsselwort 351
 - try-Block, mit catch-Phrasen verbinden 357
 - Typen von
 - Fehlern 92, 93
 - Variablen 46 - 49
 - Variablen, int Begrenzungen 42 - 45
 - Zeiger 132, 133
- U**
- überladen
 - Elementfunktionen 203 - 205
 - Elementfunktionen in Unterklassen 252
 - Funktionen 222, 223
 - Konstruktoren 221, 222
 - rein virtuelle Funktionen 272
 - überladen des Zuweisungsoperator 325, 326
 - Code 327 - 329
 - protected-Funktionen 330, 331
 - Überladen von Operatoren 311, 313, 315
 - binäre Operatoren, Verändern von Werten 318
 - Cast-Operator 321 - 323
 - Elementfunktionen, Operatoren 321
 - Funktionen, nicht Element 320
 - Funktionen Operatoren, ihre Beziehung 312
 - Funktionen, rationalize() 316
 - Heap-Speicher 317
 - Klassen, Freunde 315
 - Lesbarkeit, verbessern 311, 312
 - Links-Shift (<<)-Operator 311, 312
 - Nichtelement-Funktions 320, 321
 - Objekte, konverieren 323
 - Objekte, temporär 317
 - operator+() 315 - 318
 - operator++() Funktion 315
 - Postfix-Operator x++ 316
 - Präfix-Operator ++x 316
 - rationalize()-Funktion 316
 - Rechts-Shift (>)-Operator 311, 312
 - temporäre Objekte 317

unärer operator++(), Argumente 317, 318
 Verwendung von 311 - 313
 Zuweisungs (=)-Operator 311, 312
 überladene Operatoren
 operator<<() 333
 operator>() 333
 Überschreiben von Elementfunktionen 253
 Umlenkungssymbol (<) 153
 Umlenkungssymbol (>) 153
 unäre Operatorer 55 - 58
 unärer Operator++(), Argumente 317, 318
 unäres * Zeichen 130
 Und-Zeichen (&) 125
 Uneindeutigkeit, Mehrfachvererbung 278, 279
 Uneindeutigkeit vermeiden 34
 Ungleichheits-Operator (!=) 60
 universelle Zeiger 161
 Unterklassen
 abstrakte 271, 272
 Elementfunktionen überladen 252
 fstream 334 - 337
 konkrete, rein virtuelle Funktionen implementiert
 273
 konstruieren 246 - 249
 stringstream 337 - 341
 Unterklassen 189
 Unterscheidung Groß- und Kleinschreibung 9,
 32
 Unterstrich und einfache Hochkommata ('_') 34
 USDollar-Programm
 Code 319, 320
 Code zur Konvertierung in Zeichenkette für Aus-
 gabe 338 - 340
 Insertter, Code für Ausgabe 343, 344
 USDollar::output(), ostream Klasse, Code
 340;341
 USDollarAdd Programm-Code 313 - 315
 USDollarCast Programm-Code 321 - 323
 User Screen-Kommando (Windows Menü) 27

V

Variablen 41
 ansehen 175 - 179
 auto 90
 benennen 33, 51
 C++-Programme 32, 33
 char 47
 dezimale Zahlen 42 - 46
 double 47
 double, Speicher 128
 float, Speicher 128
 gemischte Ausdrücke 50
 Gleitkomma Begrenzungen 46
 int-Variablentyp, Begrenzungen 42 - 45
 int, Speicher 128
 Konstanten 48
 logische 62
 lokale ansehen 179, 180
 long 47
 long, Speicher 128
 mischen 50, 51
 modifizieren 176 - 179
 nGlobal 90
 nLocal 90
 nLoopCount Variable 72
 nStatic 90
 Schleifen 77
 Sonderzeichen 48, 49
 speichern 90
 Typen 46 - 49
 von Zeiger 129, 130, 132
 Werte 99
 Zeichenketten 47
 Variablen speichern 90
 Variablen Window 179, 180
 Vererbung 252
 Basisklassen, konstruieren 248
 Basisklassen, vernichten 248
 Faktorisieren 263 - 266
 GraduateStudent-Objekt HAS_A Beziehung 249,
 250
 implementieren 242 - 245
 Polymorphie 255
 public-Schlüsselwort 245
 Unterklassen, konstruieren 246 - 249
 virtual 279 - 285
 Vorteile 241

418 *Index*

- Vererbung, mehrfache 276 - 278
 - Nachteile 285, 286
 - SleeperSofa, Speicheranordnung 281, 282, 284
 - Uneindeutigkeit 278, 279
 - virtuelle Vererbung 279 - 285
 - verkettete Liste 160, 161, 163
 - Eigenschaften 164
 - Elemente am Kopf einfügen 161
 - Elemente ans Ende anfügen 162, 163
 - Anwendung, debugging 176
 - verkettete Zeiger, nicht initialisiert 167
 - Vernichten von Objekten 206 - 213
 - vertrauenswürdige Funktionen, definiert 217
 - Verzeichnis, aktuelles 295
 - Verzeichnisse, BIN 19
 - Verzweigungs-Kommando oder Anweisung 69 - 71
 - View Menü-Kommandos
 - Arbeitsbereich 294
 - Debug-Fenster 179, 180
 - View User Screen-Kommando 169
 - View Variable-Kommando 169
 - virtuelle Vererbung 279 - 285
 - virtual Destruktoren 261, 262
 - virtual display()-Elementfunktion, schlaue Inserter 344
 - virtual Funktionen 259, 260
 - inline 261
 - rein, Syntax 267, 268
 - Zeiger 261
 - virtual Konstruktoren 261
 - virtual Schlüsselwort 257, 258, 284
 - VirtuallInserter Programm-Code 344 - 346
 - Visual C++ 8
 - Arbeitsbereich einrichten 11
 - Debugger 178 - 180
 - Fehlermeldungen 14, 95, 96
 - Hilfe 16, 17
 - Installation 8
 - nNums 96
 - Projektdateien anlegen 294, 295
 - void-Funktion, return-Anweisung 85
 - void-Konstruktoren 209, 222, 223
 - void-Schlüsselwort 84
 - void strcat(target source)-Funktion 116
 - Vorrang von Operatoren 55, 56
 - Vorteile von Funktionen 84, 85
- ## W
- Web-Sites, Delorie 19
 - Werte
 - Argumente, Default-Werte 223
 - filebuf::openprot 334
 - filebuf::sh_none 334
 - filebuf::sh_read 334
 - filebuf::sh_write 334
 - von Variablen 100
 - while-Anweisung, breakpoint Kommando 175
 - WhileDemo Programm-Code 71, 72
 - while-Schleife 71, 72, 74, 115
 - wide characters 117
 - width-Parameter 342
 - width()-Funktion 342
 - Windows-Menü, Kommandos, User Screen 27
 - Windows-Programme, entwickeln mit GNU C++ 27
 - WRITE-Anweisung 92, 93
 - WRITE-Anweisung, Zeigerfehler 170
 - WWW (World Wide Web), GNU C++ installieren von 19, 21
- ## X
- XOR
 - bitweiser Operator (~) 64
 - Operator 64
- ## Z
- Zahlen
 - binäre 62, 63

- dezimale 42 - 46
- Division-vor-Addition Algorithmus 44
- Gleitkomma 45
- hexadezimal 63, 66
- Integer, gebrochene Werte 42
- Integer, Abschneiden 44, 45
- Integer, Lösen des Abschneideproblems 45, 46
- Integer, Runden 42 - 45
- mitteln, Code 43
- Rundungsfehler 46
- Zeichen 101
 - Arrays von 111 - 113, 145 - 147
 - spezielle, Variablen 48, 49
 - Variablen, Namengebung 51
 - wide 117
 - Zeichen-Begrenzung, überprüfen 121, 122
 - Zeichenketten, Arrays 151, 152
- Zeichen-Grenzen überprüfen 121, 122
- Zeichenketten
 - behandeln 338 - 341
 - benennen 113
 - definieren 113
 - Funktionen zur Manipulation 116, 117
 - initialisieren 113
 - manipulieren 114 - 117
 - mit null terminieren 177
 - terminieren 115
 - wide characters 117
 - Ziel 177
- Zeichenketten I/O, USDollar Programm-Code,
 Konvertierung in Zeichenkette für Ausgabe 338 -
 340
- Zeichenketten, Variablen 47
- Zeichen-Limit, überprüfen 121, 122
- Zeiger 127, 140
 - Argumente über Projekteinstellung übergeben
 154
 - Array-basierte Manipulation von Zeichenketten
 145, 146
 - Arrays 140 - 147
 - Arrays, Datenstruktur 159, 160
 - Arrays, Unterschiede zwischen 148 - 150
 - auf Objekte 156 - 159
 - Container FIFO (first-in-first-out) 168
 - double Variable, Speicher 128
 - Elementfunktionen, aufrufen 201, 202
 - Fehler, Schreibenanweisung 170
 - float-Variable, Speicher 128
 - Geltungsbereich der Variable 135 - 138
 - Heap-Speicher 135 - 138, 159
 - initialisieren 161
 - int-Variable, Speicher 128
 - linkbare Klassen, deklarieren 160
 - long-Variable, Speicher 128
 - Objekte übergeben 157 - 159
 - Offsets und Arrays 142
 - Operationen auf Typen 148
 - Operationen definiert auf Typen 140, 141
 - Operatoren 129
 - Projekteinstellungen, Argumente übergeben 154
 - Referenzen 158, 159
 - Speicheradressen 128, 129
 - Typ von 132, 133
 - überwachen 176
 - universaler 161
 - Variablen 129 - 131
 - Variablen p (Präfix) 130
 - Variablen, Argumente an Funktionen übergeben
 133 - 135
 - verkettet, nicht initialisiert 167
 - virtuelle Funktionen 261
 - Zeichenarrays 145 - 147
- Zeiger + Offset-Operation 140, 141
- Zeiger-Offset-Operation 140, 141
- Zeiger überwachen 176
- Zeiger, verkettete Listen 160, 163
 - Eigenschaften 164
 - Elemente am Ende-Zeiger anfügen 162, 163
 - Objekte am Kopf-Zeiger einfügen 162
- Zeiger2 Zeiger1-Operation 140, 141
- Zeilen einrücken 9
- Zielzeichenkette 177
- Ziffer, definiert 62
- zip-Dateien 19
- ZIP-Picker 19
- Zugriff auf
 - Eigenschaften von Objekten, Syntax 123
 - Elemente mit Elementfunktion 201 - 203
 - nArray 108
- Zugriff auf Objekte kontrollieren 214 - 218
- Zugriffsfunktion, definiert 216, 217

420 *Index*

Zugriffskontrolle, Objekte 214, 217, 218

private Schlüsselwort 216

protected Schlüsselwort 214 - 217

public Schlüsselwort 214

Zuweisungs (=)-Operator 311, 312

Zuweisungsoperator 34, 57, 58

Code zum Überladen 327 - 329

Elementfunktion operator=() 330

Name& (return type), operator=() 329, 330

operator=(), Default-Definition 325, 326

operator=(), Elementfunktion 330

operator=(), Name& (return type) 329, 330

überladen 325 - 330

überladen mit geschützten Funktionen 330, 331

Vergleich mit Kopierkonstruktor 326, 327

Zweige #else 302

GNU General Public License

Die folgende deutsche Übersetzung wurde im Auftrag der SuSE GmbH [suse@suse.de] von Katja Lachmann Übersetzungen [na194@fim.uni-erlangen.de] erstellt und von Peter Gerwinski [peter.gerwinski@uni-essen.de] (31. Oktober 1996) modifiziert. Diese Übersetzung wird mit der Absicht angeboten, das Verständnis der GNU General Public License (GNU-GPL) zu erleichtern. Es handelt sich jedoch nicht um eine offizielle oder im rechtlichen Sinne anerkannte Übersetzung.

Die Free Software Foundation (FSF) ist nicht der Herausgeber dieser Übersetzung, und sie hat diese Übersetzung auch nicht als rechtskräftigen Ersatz für die Original-GNU GPL anerkannt. Da die Übersetzung nicht sorgfältig von Anwälten überprüft wurde, können die Übersetzer nicht garantieren, dass die Übersetzung die rechtlichen Aussagen der GNU GPL exakt wiedergibt. Wenn Sie sicher gehen wollen, dass von Ihnen geplante Aktivitäten im Sinne der GNU GPL gestattet sind, halten Sie sich bitte an die englischsprachige Originalversion. Die Free Software Foundation möchte Sie darum bitten, diese Übersetzung nicht als offizielle Lizenzbedingungen für von Ihnen geschriebene Programme zu verwenden. Bitte benutzen Sie hierfür stattdessen die von der Free Software Foundation herausgegebene englischsprachige Originalversion.

GNU General Public License.

Deutsche Übersetzung der Version 2, Juni 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc. 675 Mass Ave, Cambridge, MA 02139, USA

Jeder hat das Recht, diese Lizenzurkunde zu vervielfältigen und unveränderte Kopien zu verbreiten; Änderungen sind jedoch nicht gestattet. Diese Übersetzung ist kein rechtskräftiger Ersatz für die englischsprachige Originalversion!

Vorwort

Die meisten Softwarelizenzen sind daraufhin entworfen worden, Ihnen die Freiheit zu nehmen, Software weiterzugeben und zu verändern. Im Gegensatz dazu soll Ihnen die GNU General Public License, die allgemeine öffentliche GNU-Lizenz, eben diese Freiheit garantieren. Sie soll sicherstellen, dass die Software für alle Benutzer frei ist. Diese Lizenz gilt für den Großteil der von der Free Software Foundation herausgegebenen Software und für alle anderen Programme, deren Autoren ihr Werk dieser Lizenz unterstellt haben. Auch Sie können diese Möglichkeit der Lizenzierung für Ihre Programme anwenden. (Ein anderer Teil der Software der Free Software Foundation unterliegt statt dessen der GNU Library General Public License, der allgemeinen öffentlichen GNU-Lizenz für Bibliotheken.)

422 GNU General Public License

Die Bezeichnung »freie« Software bezieht sich auf Freiheit, nicht auf den Preis. Unsere Lizenzen sollen Ihnen die Freiheit garantieren, Kopien freier Software zu verbreiten (und etwas für diesen Service zu berechnen, wenn Sie möchten), die Möglichkeit, die Software im Quelltext zu erhalten oder den Quelltext auf Wunsch zu bekommen. Die Lizenzen sollen garantieren, dass Sie die Software ändern oder Teile davon in neuen freien Programmen verwenden dürfen – und dass Sie wissen, dass Sie dies alles tun dürfen.

Um Ihre Rechte zu schützen, müssen wir Einschränkungen machen, die es jedem verbietet, Ihnen diese Rechte zu verweigern oder Sie aufzufordern, auf diese Rechte zu verzichten. Aus diesen Einschränkungen folgen bestimmte Verantwortlichkeiten für Sie, wenn Sie Kopien der Software verbreiten oder sie verändern.

Beispielsweise müssen Sie den Empfängern alle Rechte gewähren, die Sie selbst haben, wenn Sie – kostenlos oder gegen Bezahlung – Kopien eines solchen Programms verbreiten. Sie müssen sicherstellen, dass auch sie den Quelltext erhalten bzw. erhalten können. Und Sie müssen ihnen diese Bedingungen zeigen, damit sie ihre Rechte kennen.

Wir schützen Ihre Rechte in zwei Schritten: (1) Wir stellen die Software unter ein Urheberrecht (Copyright), und (2) wir bieten Ihnen diese Lizenz an, die Ihnen das Recht gibt, die Software zu vervielfältigen, zu verbreiten und/oder zu verändern.

Um die Autoren und uns zu schützen, wollen wir darüber hinaus sicherstellen, dass jeder erfährt, dass für diese freie Software keinerlei Garantie besteht. Wenn die Software von jemand anderem modifiziert und weitergegeben wird, möchten wir, dass die Empfänger wissen, dass sie nicht das Original erhalten haben, damit von anderen verursachte Probleme nicht den Ruf des ursprünglichen Autors schädigen.

Schließlich und endlich ist jedes freie Programm permanent durch Software-Patente bedroht. Wir möchten die Gefahr ausschließen, dass Distributoren eines freien Programms individuell Patente lizenzieren – mit dem Ergebnis, dass das Programm proprietär würde. Um dies zu verhindern, haben wir klargestellt, dass jedes Patent entweder für freie Benutzung durch jedermann lizenziert werden muss oder überhaupt nicht lizenziert werden darf.

Es folgen die genauen Bedingungen für die Vervielfältigung, Verbreitung und Bearbeitung:

Bedingungen für die Vervielfältigung, Verbreitung, und Bearbeitung.

Paragraph 0

Diese Lizenz gilt für jedes Programm und jedes andere Werk, in dem ein entsprechender Vermerk des Copyright-Inhabers darauf hinweist, dass das Werk unter den Bestimmungen dieser General Public License verbreitet werden darf. Im Folgenden wird jedes derartige Programm oder Werk als »das Programm« bezeichnet; die Formulierung »auf dem Programm basierendes Werk« bezeichnet das Programm sowie jegliche Bearbeitung des Programms im urheberrechtlichen Sinne, also ein Werk, welches das Programm, auch auszugsweise, sei es unverändert oder verändert und/oder in eine andere Sprache übersetzt, enthält. (Im Folgenden wird die Übersetzung ohne Einschränkung als »Bearbeitung« eingestuft.) Jeder Lizenznehmer wird im Folgenden als »Sie« angesprochen.

Andere Handlungen als Vervielfältigung, Verbreitung und Bearbeitung werden von dieser Lizenz nicht berührt; sie fallen nicht in ihren Anwendungsbereich. Der Vorgang der Ausführung des Programms wird nicht eingeschränkt, und die Ausgaben des Programms unterliegen dieser Lizenz nur, wenn der Inhalt ein auf dem Programm basierendes Werk darstellt (unabhängig davon, dass die Ausgabe durch die Ausführung des Programmes erfolgte). Ob dies zutrifft, hängt von den Funktionen des Programms ab.

Paragraph 1

Sie dürfen auf beliebigen Medien unveränderte Kopien des Quelltextes des Programms, wie Sie ihn erhalten haben, anfertigen und verbreiten. Voraussetzung hierfür ist, dass Sie mit jeder Kopie einen entsprechenden Copyright-Vermerk sowie einen Haftungsausschluss veröffentlichen, alle Vermerke, die sich auf diese Lizenz und das Fehlen einer Garantie beziehen, unverändert lassen und des Weiteren allen anderen Empfängern des Programms zusammen mit dem Programm eine Kopie dieser Lizenz zukommen lassen.

Sie dürfen für den eigentlichen Kopiervorgang eine Gebühr verlangen. Wenn Sie es wünschen, dürfen Sie auch gegen Entgelt eine Garantie für das Programm anbieten.

Paragraph 2

Sie dürfen Ihre Kopie(n) des Programms oder eines Teils davon verändern, wodurch ein auf dem Programm basierendes Werk entsteht; Sie dürfen derartige Bearbeitungen unter den Bestimmungen von Paragraph 1 vervielfältigen und verbreiten, vorausgesetzt, dass zusätzlich alle folgenden Bedingungen erfüllt werden:

- (a) Sie müssen die veränderten Dateien mit einem auffälligen Vermerk versehen, der auf die von Ihnen vorgenommene Modifizierung und das Datum jeder Änderung hinweist.
- (b) Sie müssen dafür sorgen, dass jede von Ihnen verbreitete oder veröffentlichte Arbeit, die ganz oder teilweise von dem Programm oder Teilen davon abgeleitet ist, Dritten gegenüber als Ganzes unter den Bedingungen dieser Lizenz ohne Lizenzgebühren zur Verfügung gestellt wird.
- (c) Wenn das veränderte Programm normalerweise bei der Ausführung interaktiv Kommandos einliest, müssen Sie dafür sorgen, dass es, wenn es auf dem üblichsten Wege für solche interaktive Nutzung gestartet wird, eine Meldung ausgibt oder ausdrückt, die einen geeigneten Copyright-Vermerk enthält sowie einen Hinweis, dass es keine Gewährleistung gibt (oder anderenfalls, dass Sie Garantie leisten), und dass die Benutzer das Programm unter diesen Bedingungen weiter verbreiten dürfen. Auch muss der Benutzer darauf hingewiesen werden, wie eine Kopie dieser Lizenz ansehen kann. (Ausnahme: Wenn das Programm selbst interaktiv arbeitet, aber normalerweise keine derartige Meldung ausgibt, muss Ihr auf dem Programm basierendes Werk auch keine solche Meldung ausgeben).

Diese Anforderungen betreffen das veränderte Werk als Ganzes. Wenn identifizierbare Abschnitte des Werkes nicht von dem Programm abgeleitet sind und vernünftigerweise selbst als unabhängige und eigenständige Werke betrachtet werden können, dann erstrecken sich diese Lizenz und ihre Bedingungen nicht auf diese Abschnitte, wenn sie als eigenständige Werke verbreitet werden. Wenn Sie jedoch dieselben Abschnitte als Teil eines Ganzen verbreiten, das ein auf dem Programm basierendes Werk darstellt, dann muss die Verbreitung des Ganzen nach den Bedingungen dieser Lizenz erfolgen, deren Bedingungen für weitere Lizenznehmer somit auf die Gesamtheit ausgedehnt werden – und damit auf jeden einzelnen Teil, unabhängig vom jeweiligen Autor.

424 GNU General Public License

Somit ist es nicht die Absicht dieses Abschnittes, Rechte für Werke in Anspruch zu nehmen oder zu beschneiden, die komplett von Ihnen geschrieben wurden; vielmehr ist es die Absicht, die Rechte zur Kontrolle der Verbreitung von Werken, die auf dem Programm basieren oder unter seiner auszugswweisen Verwendung zusammengestellt worden sind, auszuüben.

Ferner bringt ein einfaches Zusammenstellen eines anderen Werkes, das nicht auf dem Programm basiert, zusammen mit dem Programm oder einem auf dem Programm basierenden Werk auf ein- und demselben Speicher- oder Vertriebsmedium nicht in den Anwendungsbereich dieser Lizenz.

Paragraph 3

Sie dürfen das Programm (oder ein darauf basierendes Werk gemäß Paragraph 2) als Objektcode oder in ausführbarer Form unter den Bedingungen von Paragraph 1 und 2 vervielfältigen und verbreiten – vorausgesetzt, dass Sie außerdem eine der folgenden Leistungen erbringen:

- (a) Liefern Sie das Programm zusammen mit dem vollständigen zugehörigen maschinenlesbaren Quelltext auf einem für den Datenaustausch üblichen Medium aus, wobei die Verteilung unter den Bedingungen der Paragraphen 1 und 2 erfolgen muss. Oder:
- (b) Liefern Sie das Programm zusammen mit einem mindestens drei Jahre lang gültigen schriftlichen Angebot aus, jedem Dritten eine vollständige maschinenlesbare Kopie des Quelltextes zur Verfügung zu stellen – zu nicht höheren Kosten als denen, die durch den physischen Kopiervorgang anfallen -, wobei der Quelltext unter den Bedingungen der Paragraphen 1 und 2 auf einem für den Datenaustausch üblichen Medium weitergegeben wird. Oder:
- (c) Liefern Sie das Programm zusammen mit dem schriftlichen Angebot der Zurverfügungstellung des Quelltextes aus, das Sie selbst erhalten haben. (Diese Alternative ist nur für nicht-kommerzielle Verbreitung zulässig und nur, wenn Sie das Programm als Objektcode oder in ausführbarer Form mit einem entsprechenden Angebot gemäß Absatz b erhalten haben.)

Unter dem Quelltext eines Werkes wird diejenige Form des Werkes verstanden, die für Bearbeitungen vorzugsweise verwendet wird. Für ein ausführbares Programm bedeutet »der komplette Quelltext«: Der Quelltext aller im Programm enthaltenen Module einschließlich aller zugehörigen Modulschnittstellen-Definitionsdateien sowie der zur Kompilation und Installation verwendeten Skripte. Als besondere Ausnahme jedoch braucht der verteilte Quelltext nichts von dem zu enthalten, was üblicherweise (entweder als Quelltext oder in binärer Form) zusammen mit den Hauptkomponenten des Betriebssystems (Kernel, Compiler usw.) geliefert wird, unter dem das Programm läuft – es sei denn, diese Komponente selbst gehört zum ausführbaren Programm.

Wenn die Verbreitung eines ausführbaren Programms oder von Objektcode dadurch erfolgt, dass der Kopierzugriff auf eine dafür vorgesehene Stelle gewährt wird, so gilt die Gewährung eines gleichwertigen Zugriffs auf den Quelltext als Verbreitung des Quelltextes, auch wenn Dritte nicht dazu gezwungen sind, den Quelltext zusammen mit dem Objektcode zu kopieren.

Paragraph 4

Sie dürfen das Programm nicht vervielfältigen, verändern, weiter lizenzieren oder verbreiten, sofern es nicht durch diese Lizenz ausdrücklich gestattet ist. Jeder anderweitige Versuch der Vervielfältigung, Modifizierung, Weiterlizenzierung und Verbreitung ist nichtig und beendet automatisch Ihre Rechte unter dieser Lizenz. Jedoch werden die Lizenzen Dritter, die von Ihnen Kopien oder Rechte unter dieser Lizenz erhalten haben, nicht beendet, solange diese die Lizenz voll anerkennen und befolgen.

Paragraph 5

Sie sind nicht verpflichtet, diese Lizenz anzunehmen, da Sie sie nicht unterzeichnet haben. Jedoch gibt Ihnen nichts anderes die Erlaubnis, das Programm oder von ihm abgeleitete Werke zu verändern oder zu verbreiten. Diese Handlungen sind gesetzlich verboten, wenn Sie diese Lizenz nicht anerkennen. Indem Sie das Programm (oder ein darauf basierendes Werk) verändern oder verbreiten, erklären Sie Ihr Einverständnis mit dieser Lizenz und mit allen ihren Bedingungen bezüglich der Vervielfältigung, Verbreitung und Veränderung des Programms oder eines darauf basierenden Werks.

Paragraph 6

Jedes Mal, wenn Sie das Programm (oder ein auf dem Programm basierendes Werk) weitergeben, erhält der Empfänger automatisch vom ursprünglichen Lizenzgeber die Lizenz, das Programm entsprechend den hier festgelegten Bestimmungen zu vervielfältigen, zu verbreiten und zu verändern. Sie dürfen keine weiteren Einschränkungen der Durchsetzung der hierin zugestandenen Rechte des Empfängers vornehmen. Sie sind nicht dafür verantwortlich, die Einhaltung dieser Lizenz durch Dritte durchzusetzen.

Paragraph 7

Sollten Ihnen infolge eines Gerichtsurteils, des Vorwurfs einer Patentverletzung oder aus einem anderen Grunde (nicht auf Patentfragen begrenzt) Bedingungen (durch Gerichtsbeschluss, Vergleich oder anderweitig) auferlegt werden, die den Bedingungen dieser Lizenz widersprechen, so befreien Sie diese Umstände nicht von den Bestimmungen dieser Lizenz. Wenn es Ihnen nicht möglich ist, das Programm unter gleichzeitiger Beachtung der Bedingungen in dieser Lizenz und Ihrer anderweitigen Verpflichtungen zu verbreiten, dann dürfen Sie als Folge das Programm überhaupt nicht verbreiten. Wenn zum Beispiel ein Patent nicht die gebührenfreie Weiterverbreitung des Programms durch diejenigen erlaubt, die das Programm direkt oder indirekt von Ihnen erhalten haben, dann besteht der einzige Weg, sowohl das Patentrecht als auch diese Lizenz zu befolgen, darin, ganz auf die Verbreitung des Programms zu verzichten.

Sollte sich ein Teil dieses Paragraphen als ungültig oder unter bestimmten Umständen nicht durchsetzbar erweisen, so soll dieser Paragraph seinem Sinne nach angewandt werden; im übrigen soll dieser Paragraph als Ganzes gelten.

Zweck dieses Paragraphen ist nicht, Sie dazu zu bringen, irgendwelche Patente oder andere Eigentumsansprüche zu verletzen oder die Gültigkeit solcher Ansprüche zu bestreiten; dieser Paragraph hat einzig den Zweck, die Integrität des Vertriebungssystems der freien Software zu schützen, das durch die Praxis öffentlicher Lizenzen verwirklicht wird. Viele Leute haben großzügige Beiträge zu dem großen Angebot der mit diesem System verbreiteten Software im Vertrauen auf die konsistente Anwendung dieses Systems geleistet; es liegt am Autor/Geber zu entscheiden, ob er die Software mittels irgendeines anderen Systems verbreiten will; ein Lizenznehmer hat auf diese Entscheidung keinen Einfluss.

Dieser Paragraph ist dazu gedacht, deutlich klarzustellen, was als Konsequenz aus dem Rest dieser Lizenz betrachtet wird.

426 GNU General Public License**Paragraph 8**

Wenn die Verbreitung und/oder die Benutzung des Programms in bestimmten Staaten entweder durch Patente oder durch urheberrechtlich geschützte Schnittstellen eingeschränkt ist, kann der Urheberrechtsinhaber, der das Programm unter diese Lizenz gestellt hat, eine explizite geografische Begrenzung der Verbreitung angeben, in der diese Staaten ausgeschlossen werden, sodass die Verbreitung nur innerhalb und zwischen den Staaten erlaubt ist, die nicht ausgeschlossen sind. In einem solchen Fall beinhaltet diese Lizenz die Beschränkung, als wäre sie in diesem Text niedergeschrieben.

Paragraph 9

Die Free Software Foundation kann von Zeit zu Zeit überarbeitete und/oder neue Versionen der General Public License veröffentlichen. Solche neuen Versionen werden vom Grundprinzip her der gegenwärtigen entsprechen, können aber im Detail abweichen, um neuen Problemen und Anforderungen gerecht zu werden.

Jede Version dieser Lizenz hat eine eindeutige Versionsnummer. Wenn in einem Programm angegeben wird, dass es dieser Lizenz in einer bestimmten Versionsnummer oder »jeder späteren Version« (»any later version«) unterliegt, so haben Sie die Wahl, entweder den Bestimmungen der genannten Version zu folgen oder denen jeder beliebigen späteren Version, die von der Free Software Foundation veröffentlicht wurde. Wenn das Programm keine Versionsnummer angibt, können Sie eine beliebige Version wählen, die je von der Free Software Foundation veröffentlicht wurde.

Paragraph 10

Wenn Sie den Wunsch haben, Teile des Programms in anderen freien Programmen zu verwenden, deren Bedingungen für die Verbreitung andere sind, schreiben Sie an den Autor, um ihn um die Erlaubnis zu bitten. Für Software, die unter dem Copyright der Free Software Foundation steht, schreiben Sie an die Free Software Foundation; wir machen zu diesem Zweck gelegentlich Ausnahmen. Unsere Entscheidung wird von den beiden Zielen geleitet werden, zum einen den freien Status aller von unserer freien Software abgeleiteten Werke zu erhalten und zum anderen das gemeinschaftliche Nutzen und Wiederverwenden von Software im Allgemeinen zu fördern.

Keine Gewährleistung**Paragraph 11**

Da das Programm ohne jegliche Kosten lizenziert wird, besteht keinerlei Gewährleistung für das Programm, soweit dies gesetzlich zulässig ist. Sofern nicht anderweitig schriftlich bestätigt, stellen die Copyright-Inhaber und/oder Dritte das Programm so zur Verfügung, »wie es ist«, ohne irgendeine Gewährleistung, weder ausdrücklich noch implizit, einschließlich – aber nicht begrenzt auf – Marktreife oder Verwendbarkeit für einen bestimmten Zweck. Das volle Risiko bezüglich Qualität und Leistungsfähigkeit des Programms liegt bei Ihnen. Sollte sich das Programm als fehlerhaft herausstellen, liegen die Kosten für notwendigen Service, Reparatur oder Korrektur bei Ihnen.

Paragraph 12

In keinem Fall, außer wenn durch geltendes Recht gefordert oder schriftlich zugesichert, ist irgendein Copyright-Inhaber oder irgendein Dritter, der das Programm wie oben erlaubt modifiziert oder verbreitet hat, Ihnen gegenüber für irgendwelche Schäden haftbar, einschließlich jeglicher allgemeiner oder spezieller Schäden, Schäden durch Seiteneffekte (Nebenwirkungen) oder Folgeschäden, die aus der Benutzung des Programms oder der Unbenutzbarkeit des Programms folgen (einschließlich – aber nicht beschränkt auf – Datenverluste, fehlerhafte Verarbeitung von Daten, Verluste, die von Ihnen oder anderen getragen werden müssen, oder dem Unvermögen des Programms, mit irgendeinem anderen Programm zusammenzuarbeiten), selbst wenn ein Copyright-Inhaber oder Dritter über die Möglichkeit solcher Schäden unterrichtet worden war.

