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.*

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 kg 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 Gleitkommavariablen gleich sind.

- *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 geshiftet, 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.*

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 ist 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 Rückgabetyp nicht Teil des Namens ist.*

Samstagnachmittag

1.

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

2.

```
void readAndDisplay()
{
    Student s;

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

    // Ausgabe der Studenteninformationen
    cout << »\nStudenteninformationen:\n«;
    cout << s.szLastName << »\n«;
    cout << s.nGrade << »\n«;
    cout << s.dAverage << »\n«;
}
```

3.

```
void readAndDisplayAverage()
{
    Student s;

    // Eingabe Studenteninformationen
    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 Studenteninformationen
    cout << »\nStudenteninformationen:\n«;
    cout << s.szLastName << »\n«;
    cout << s.nGrade << »\n«;
    cout << s.dAverage << »\n«;
}
```

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.

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);
}
```

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
    // letzen 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*** `return-Previous(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.***

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.***

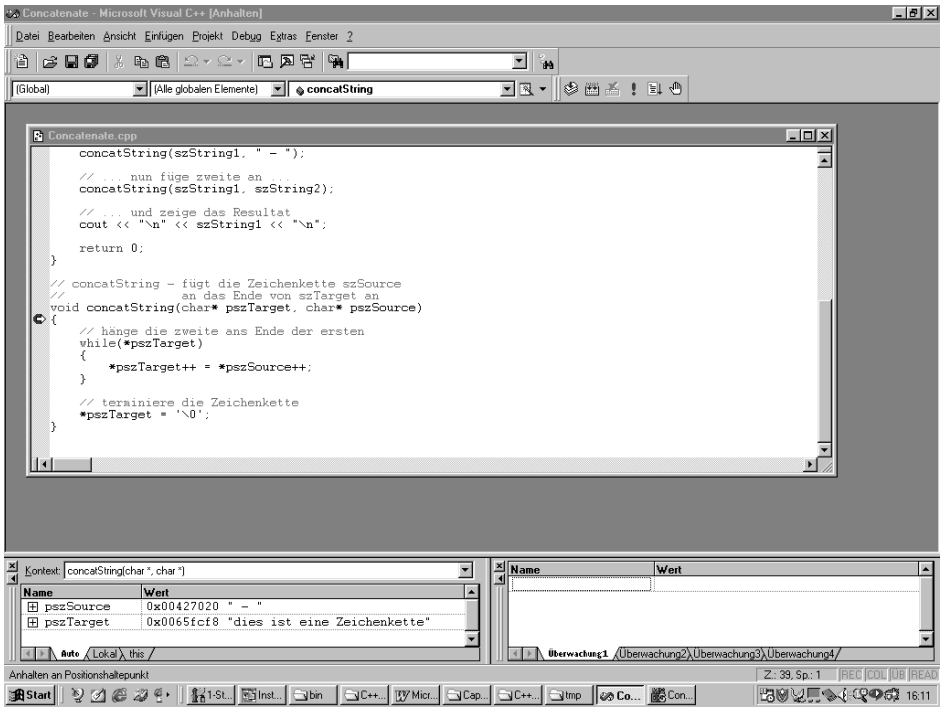

Abbildung A.1: Das Fenster zeigt die Quellzeichenkette und die Zielzeichenkette an.

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.*

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

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.***

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];
```

```
// 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.***

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)
    {
```

```
        // 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
{
```

```
  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`***?***

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
```

```
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.*

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 Rückgabetyp 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
```

```
        // 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 wirk 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
```

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 Rükkgaben 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 : pubic Vehicle
{
  public:
    Car(int nCyliners, 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++.

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 ToStringWOStream.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 ToStringWStream.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

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

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 sichergehen 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

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.)

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 Entgeld 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 ausdruckt, 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.

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 auszugsweisen 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 Verbreitungssystems 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.

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.