

SCHETT MATTHIAS

# SEN-ÜBUNG 05

# *Inhaltsverzeichnis*

*Aufgabe 1*      3

*Lösungsidee*      3

*Testfälle*      5

*Aufgabe 2*      6

*Lösungsidee*      6

*Testfälle*      7

*Anhang A: Aufgabe 1*      8

*Anhang B: Aufgabe 2*      14

# Aufgabe 1

## Lösungsidee

Um ein Wertepaar abzuspeichern wurde eine Struktur namens DiveData erstellt. Diese Struktur wird dann in einem std::vector abgespeichert.

```
1 struct DiveData {
2     time_t mTime;
3     double mDepth;
4     double mUpDown;
5 };
6
7 extern std::vector<DiveData> diveComputer;
```

Zum Abspeichern des Eingabestromes wird der scanner Klasse ein Eingabestrom übergeben und dieser anschließend durchgegangen und gelesen, sollten die Werte nicht den Erwartungen entsprechen wird eine std::exception mit der Meldung <Unknown format> ausgelöst um anzuzeigen, dass ein Fehler aufgetreten ist.

Die Ausgabe erfolgt anschließend in Tabellenform, dafür wurden die Folgenden Manipulatoren erstellt.

```
1 ostream& hr(ostream& os) {
2     return os << "_____"
3     _____";
4 }
5
6 ostream& colSpace(ostream& os) {
7     return os << setw(colSpacing) << " ";
8 }
9
10 ostream& colWidth(ostream& os){
11     return os << setw(colWidthNum) << " ";
12 }
13
14 ostream& colFormat(ostream& os) {
15     return os << setw(colWidthNum);
16 }
17
18 ostream& formatUpDown(ostream &os){
```

Das extern vor der Deklaration des Vektors ist notwendig um Mehrfach Definitionen zu verhindern

```
19     return os << right << setiosflags(ios::fixed)
20         << setprecision(4);
21 }
22 ostream& formatDepth(ostream &os){
23     return os << right << setiosflags(ios::fixed)
24         << setprecision(2);
25 }
```

Der Programmcode befindet sich ab Aufgabe 1

*Testfälle*

Listing 1: Input Testfall 1

```

1 0 (0.0) 10 (2.5) 50 (6.8) 150 (15.0) 270 (23.88)
   800 (26.0) 1235 (20.5) 1780 (15.8) 2345(8.3)
   3876 (0.0)

```

Listing 2: Input Testfall 2

```

1 0 0.0) 10 (2.5) 50 (6.8) 150 (15.0) 270 (23.88)
   800 (26.0) 1235 (20.5) 1780 (15.8) 2345(8.3)
   3876 (0.0)

```

Listing 3: Ausgabe

```

1
2
3 Testfall 01
4 -----
5
6 Dive Time      Dive Depth      Down/Up
7 (hh:mm:ss)      (m)              (m/sec)
8 -----
9 00:00:00         0.00
10                  -0.2500
11 00:00:10         2.50
12                  -0.1075
13 00:00:50         6.80
14                  -0.0820
15 00:02:30        15.00
16                  -0.0740
17 00:04:30        23.88
18                  -0.0040
19 00:13:20        26.00
20                  0.0126
21 00:20:35        20.50
22                  0.0086
23 00:29:40        15.80
24                  0.0133
25 00:39:05         8.30
26                  0.0054
27 01:04:36         0.00
28
29
30 Testfall 02
31 -----
32
33 Unkown format

```

## Aufgabe 2

### Lösungsidee

Da eine Lagerverwaltung aus einem Lager und das Lager wiederum aus Artikeln besteht, benötigen wir 2 Klassen.

- Article
- Warehouse

Diese beiden Klassen ermöglichen uns dann die Lagerverwaltung.

Ein Artikel besteht aus :

```
1  int mArticleNumber;  
2  std::string mArticleName;  
3  size_t mQuantity;  
4  double mPrice;
```

Jeder dieser Member besitzt eine Getter und eine Setter Methode die es erlaubt die Werte zu lesen und zu verändern. Weiters wurde operator< überschrieben um die Lagermenge<sup>1</sup> zu vergleichen.

<sup>1</sup> mQuantity

```
1  bool Article::operator<(Article const & vgl) const  
2  {  
3      return (mQuantity < vgl.mQuantity);  
4  }
```

Das Lager besteht aus:

```
1  std::vector<Article> mArticles;  
2  std::string mWarehouseName;
```

Hier besitzt allerdings keiner der Member eine Getter oder Setter Funktion. Es gibt nur eine GetNumberOfArticles Funktion die, die aktuelle Größe des std::vector ausgibt. Um einen neuen Artikel abzuspeichern gibt es die AddArticle(Article const &newArticle) Funktion, die std::vector::push\_back aufruft. Es gibt auch die Möglichkeit aus einem File einzulesen, dabei hilft abermals die scanner Klasse und liest das File ein, tritt bei der Verarbeitung ein Fehler auf, wenn das Format nicht richtig ist, wird eine std::exception geworfen mit der Meldung <Unknown format>. Die Ausgabe erfolgt anschließend im vorgegebenen Tabellenformat.

Der Quellcode findet sich ab Aufgabe 2

*Testfälle*

Listing 4: Input Testfall 1

```

1 4711 - "Kabelkanal (3m)" - 10 - 1.50;
2 1147 - "Installationsrohr (5m)" - 49 - 3.49;
3 7141 - "Funksteckdose" - 3 - 11.99;
4 1471 - "Wechselschalter" - 17 - 7.90;
5 1417 - "Ein-/Ausschalter" - 24 - 6.99;
6 1714 - "Zeitschaltuhr" - 5 - 33.50;

```

Listing 5: Input Testfall 2

```

1 4711 - "Kabelkanal (3m)" - 10 -;
2 1147 - "Installationsrohr (5m)" - 49 - 3.49;

```

Listing 6: Ausgabe

```

1
2
3 Testfall 01
4 -----
5
6 Article list of Warehouse TestWarehouse
7
8 7141 Funksteckdose          3    11.99
9 1714 Zeitschaltuhr          5    33.50
10 4711 Kabelkanal (3m)       10    1.50
11 1471 Wechselschalter       17    7.90
12 1417 Ein-/Ausschalter      24    6.99
13 1147 Installationsrohr (5m) 49    3.49
14
15
16 Testfall 02
17 -----
18
19 Unknown format

```

# Anhang A

## Aufgabe 1

Listing A.1: Header für den Tauchcomputer

```
1  //////////////////////////////////////
2  // Workfile      : Main.cpp
3  // Author        : Matthias Schett
4  // Date          : 12-04-2013
5  // Description   : Dive Computer
6  // Remarks       : -
7  // Revision      : 0
8  //////////////////////////////////////
9
10 #ifndef DIVE_H
11 #define DIVE_H
12
13 #include <ctime>
14 #include <vector>
15 #include <ostream>
16 #include <istream>
17 #include <exception>
18
19 int const colSpacing = 4;
20 int const colWidthNum = 10;
21 int const diveDepthPrecision = 2;
22 int const upDownPrecision = 3;
23
24 struct DiveData{
25     time_t mTime;
26     double mDepth;
27     double mUpDown;
28 };
29
30 extern std::vector<DiveData> diveComputer;
31
32 /* *****
33 // Method:    readDiveData
34 // FullName:  readDiveData
35 // Access:    public
```



```

36 // Returns: void
37 // Qualifier:
38 // Parameter: std::istream &is
39 // Reads dive data from stream and saves it –
    throws exception when an error occurs
40 //*****
41 void readDiveData(std::istream &is);
42
43 //*****
44 // Method: printDiveData
45 // FullName: printDiveData
46 // Access: public
47 // Returns: void
48 // Qualifier:
49 // Parameter: std::ostream &os
50 // Prints formatted dive data to stream
51 //*****
52 void printDiveData(std::ostream &os);
53
54 #endif

```

Listing A.2: Implementierung des Tauchcomputers

```

1 ///////////////////////////////////////////////////
2 // Workfile      : Main.cpp
3 // Author        : Matthias Schett
4 // Date          : 12-04-2013
5 // Description    : Dive Computer
6 // Remarks       : –
7 // Revision      : 0
8 ///////////////////////////////////////////////////
9
10 #include "DiveComputer.h"
11 #include <string>
12 #include "scanner.h"
13 #include <iomanip>
14
15 using namespace std;
16
17 vector<DiveData> diveComputer (0);
18
19 bool isTbReal(scanner &scan){
20     return scan.symbol_is_real();
21 }
22
23 bool isTbBracket(scanner &scan){
24     return scan.symbol_is_lpar();
25 }
26
27 bool isTbInt(scanner &scan){

```

```

28     return scan.symbol_is_integer();
29 }
30
31 time_t parseTime(scanner &scan){
32     time_t temp = scan.get_integer();
33     scan.next_symbol();
34     return temp;
35 }
36
37 double parseDepth( scanner & scan ) {
38     scan.next_symbol();
39     if(isTbReal(scan)){
40         double temp = scan.get_real();
41         scan.next_symbol();
42         return temp;
43     }
44     return 0.0;
45 }
46
47 void calcUpDown(DiveData &newData) {
48     if(!diveComputer.empty()){
49         DiveData oldData = diveComputer.at(
50             diveComputer.size() - 1);
51         newData.mUpDown = (oldData.mDepth -
52             newData.mDepth) / (newData.mTime -
53             oldData.mTime);
54     } else {
55         newData.mUpDown = 0.0;
56     }
57 }
58
59 void readDiveData( std::istream &is ){
60     scanner scan(is);
61
62     while(!scan.symbol_is_eof()){
63         if(isTbInt(scan)){
64             time_t parsedTime = parseTime(scan);
65             if(isTbBracket(scan)){
66                 double parsedDepth = parseDepth(
67                     scan);
68                 scan.next_symbol();
69                 DiveData data;
70                 data.mDepth = parsedDepth;
71                 data.mTime = parsedTime;
72                 calcUpDown(data);
73                 diveComputer.push_back(data);
74             }
75         } else {

```

```

73         throw std::exception("Unkown format");
74     }
75 }
76 }
77
78 ostream& hr(ostream& os) {
79     return os << "
    ";
80 }
81
82 ostream& colSpace(ostream& os) {
83     return os << setw(colSpacing) << " ";
84 }
85
86 ostream& colWidth(ostream& os){
87     return os << setw(colWidthNum) << " ";
88 }
89
90 ostream& colFormat(ostream& os) {
91     return os << setw(colWidthNum);
92 }
93
94 ostream& formatUpDown(ostream &os){
95     return os << right << setiosflags(ios::fixed)
        << setprecision(4);
96 }
97
98 ostream& formatDepth(ostream &os){
99     return os << right << setiosflags(ios::fixed)
        << setprecision(2);
100 }
101
102
103 void printTableHeader(std::ostream &os){
104     os << left << colFormat << "Dive Time " <<
        colSpace << colFormat << "Dive Depth " <<
        colSpace << colFormat << "Down/Up";
105     os << endl << colFormat << "(hh:mm:ss)" <<
        colSpace << colFormat << "(m)" << colSpace
        << colFormat << "(m/sec)" << endl << hr <<
        endl;
106 }
107
108 void printDataLine(std::ostream &os, DiveData
    const &data, int i){
109     struct tm * ptm = gmtime(&data.mTime);
110     if(i != 0){ // Don't print this at the first
        line
    }

```

```

111         os << colWidth << colSpace << colWidth <<
            colSpace << colFormat << formatUpDown
            << data.mUpDown << endl;
112     }
113     os << colFormat << put_time(ptm, "%H:%M%S") <<
        colSpace << colFormat << formatDepth <<
        data.mDepth << endl;
114 }
115
116 void printDiveData( std::ostream &os ){
117
118     printTableHeader(os);
119
120     for(int i = 0; i < diveComputer.size(); i++){
121         printDataLine(os, diveComputer.at(i), i);
122     }
123
124 }

```

Listing A.3: Testtreiber

```

1  //////////////////////////////////////
2  // Workfile      : Main.cpp
3  // Author       : Matthias Schett
4  // Date        : 12-04-2013
5  // Description  : Dive Computer
6  // Remarks     : -
7  // Revision    : 0
8  //////////////////////////////////////
9
10 #include <vld.h>
11 #include <iostream>
12 #include <fstream>
13 #include <string>
14 #include "scanner.h"
15 #include "DiveComputer.h"
16 #include <exception>
17
18 using namespace std;
19
20 void printTestHeader(int testNumber, ostream &
    stream){
21     stream << endl << endl;
22     if(testNumber < 10){
23         stream << "Testfall 0" << testNumber;
24     } else {
25         stream << "Testfall " << testNumber;
26     }
27     stream << endl << "—————" << endl <<
        endl;

```

```
28
29 }
30
31 int main() {
32     ofstream outFile("OutputA1.txt");
33     try {
34         printTestHeader(1, outFile);
35
36         ifstream file("Test.txt");
37         readDiveData(file);
38
39         printDiveData(outFile);
40
41         printTestHeader(2, outFile);
42
43         ifstream file2("TestIncorrect.txt");
44         readDiveData(file2);
45
46         printDiveData(outFile);
47
48         file.close();
49         file2.close();
50
51     } catch(std::exception &e) {
52         outFile << e.what();
53     }
54     outFile.close();
55     cin.get();
56     return 0;
57 }
```

# Anhang B

## Aufgabe 2

Listing B.1: Header für den Artikel

```
1  //////////////////////////////////////
2  // Workfile      : Article.h
3  // Author       : Matthias Schett
4  // Date        : 20-04-2013
5  // Description  : Ware house management
6  // Remarks     : -
7  // Revision    : 0
8  //////////////////////////////////////
9
10 #ifndef ARTICLE_H
11 #define ARTICLE_H
12
13 #include <string>
14 #include <exception>
15
16 class ArticleException: public std::exception
17 {
18
19 };
20
21 class Article {
22 private:
23
24     int mArticleNumber;
25     std::string mArticleName;
26     size_t mQuantity;
27     double mPrice;
28
29 public:
30     // Ctr
31     Article(int articleNumber, std::string
32             ArticleName, size_t quantity, double price)
33             ;
34
35     // Dtr
```

```

34     ~Article();
35
36     // Getters
37     int getArticleNumber();
38     std::string & getArticleName();
39     size_t getQuantity();
40     double getPrice();
41
42     // Setters
43     void setArticleNumber(int articleNumber);
44     void setArticleName(std::string & articleName)
45         ;
46     void setQuantity(size_t quantity);
47     // Throws exception if price is negativ
48     void setPrice(double price);
49
50     // compares the quantity of two articles
51     bool operator<(Article const & vgl) const;
52 };
53
54 #endif

```

Listing B.2: Implementierung des Artikels

```

1  //////////////////////////////////////
2  // Workfile      : Article.cpp
3  // Author       : Matthias Schett
4  // Date        : 20-04-2013
5  // Description  : Ware house management
6  // Remarks     : -
7  // Revision    : 0
8  //////////////////////////////////////
9  #include "Article.h"
10 #include <exception>
11
12 using namespace std;
13
14 Article::Article(int articleNumber, std::string
15     articleName, size_t quantity, double price) :
16     mArticleNumber(articleNumber), mArticleName(
17     articleName), mQuantity(quantity), mPrice(price
18     )
19 {
20 }
21
22 Article::~Article()
23 {
24 }

```

```

22 int Article::getArticleNumber() {
23     return mArticleNumber;
24 }
25
26 std::string &Article::getArticleName() {
27     return mArticleName;
28 }
29
30 size_t Article::getQuantity() {
31     return mQuantity;
32 }
33
34 double Article::getPrice() {
35     return mPrice;
36 }
37
38 void Article::setArticleNumber(int articleNumber) {
39     mArticleNumber = articleNumber;
40 }
41
42 void Article::setArticleName(std::string &
    articleName) {
43     mArticleName = articleName;
44 }
45
46 void Article::setQuantity(size_t quantity) {
47     mQuantity = quantity;
48 }
49
50 void Article::setPrice(double price) {
51     if(price < 0.0) {
52         throw std::exception("Price is not allowed
            to be negative");
53     }
54     mPrice = price;
55 }
56
57 bool Article::operator<(Article const & vgl) const
    {
58     return (mQuantity < vgl.mQuantity);
59 }

```

Listing B.3: Header für das Lager

```

1 //////////////////////////////////////////////////
2 // Workfile      : Warehouse.h
3 // Author        : Matthias Schett
4 // Date          : 20-04-2013
5 // Description   : Ware house management
6 // Remarks      : -

```



```

7 // Revision : 0
8 ///////////////////////////////////////////////////
9 #ifndef WAREHOUSE_H
10 #define WAREHOUSE_H
11
12 #include "Article.h"
13 #include <vector>
14 #include <fstream>
15 #include <ostream>
16
17 class Warehouse {
18 private:
19
20     std::vector<Article> mArticles;
21     std::string mWarehouseName;
22
23
24     size_t searchLongestName();
25     size_t searchHighestArticleNum();
26     size_t searchHighestQuantity();
27     size_t searchHighestPrice();
28
29 public:
30     Warehouse(std::string warehouseName);
31     ~Warehouse();
32
33     /* *****
34     // Method: addArticle
35     // FullName: Warehouse::addArticle
36     // Access: public
37     // Returns: void
38     // Qualifier:
39     // Parameter: Article const & newArticle
40     //*****
41     void addArticle(Article const & newArticle);
42
43     //*****
44     // Method: getNumberOfArticles
45     // FullName: Warehouse::getNumberOfArticles
46     // Access: public
47     // Returns: size_t
48     // Qualifier:
49     //*****
50     size_t getNumberOfArticles();
51
52     //*****
53     // Method: readArticlesFromFile
54     // FullName: Warehouse::readArticlesFromFile
55     // Access: public

```

```

56 // Returns: void
57 // Qualifier:
58 // Parameter: ifstream & file
59 // Reads articles from a file and adds them
60 //*****
61 void readArticlesFromFile(std::ifstream &file)
62     ;
63 //*****
64 // Method: printArticleList
65 // FullName: Warehouse::printArticleList
66 // Access: public
67 // Returns: void
68 // Qualifier:
69 // Parameter: ostream & os
70 // Prints the articles to the given stream
71 //*****
72 void printArticleList(std::ostream &os);
73 };
74 #endif // WAREHOUSE_H

```

Listing B.4: Implementierung des Lagers

```

1 ///////////////////////////////////////////////////
2 // Workfile      : Warehouse.cpp
3 // Author        : Matthias Schett
4 // Date          : 20-04-2013
5 // Description   : Ware house management
6 // Remarks       : -
7 // Revision      : 0
8 ///////////////////////////////////////////////////
9 #include "WareHouse.h"
10 #include "scanner.h"
11 #include <exception>
12 #include <algorithm>
13 #include <iomanip>
14
15 using namespace std;
16
17 // Comparison function object for sort method
18 class comp
19 {
20 public:
21     bool operator() (const Article &a, const
22                     Article &b) const
23     {
24         return a < b;
25     }
26 };

```

```

27 int intlen(float start) {
28     int end = 0;
29     while(start >= 1) {
30         start = start/10;
31         end++;
32     }
33     return end;
34 }
35
36 size_t Warehouse::searchLongestName() {
37     size_t length = 0;
38     for (size_t i = 0; i < mArticles.size() - 1; i
39         ++){
40         size_t length1 = mArticles.at(i).
41             getArticleName().length();
42         size_t length2 = mArticles.at(i+1).
43             getArticleName().length();
44         if(length1 < length2){
45             length = length2;
46         } else if(length1 > length2){
47             length = length1;
48         }
49     }
50
51     return length;
52 }
53
54 size_t Warehouse::searchHighestArticleNum() {
55     size_t length = 0;
56     for (size_t i = 0; i < mArticles.size() - 1; i
57         ++){
58         size_t length1 = mArticles.at(i).
59             getArticleNumber();
60         size_t length2 = mArticles.at(i+1).
61             getArticleNumber();
62         if(length1 < length2){
63             length = length2;
64         } else if(length1 > length2){
65             length = length1;
66         }
67     }
68
69     return intlen(length);
70 }
71
72 size_t Warehouse::searchHighestQuantity() {
73     size_t length = 0;
74     for (size_t i = 0; i < mArticles.size() - 1; i
75         ++){

```

```

69         size_t length1 = mArticles.at(i).
           getQuantity();
70         size_t length2 = mArticles.at(i+1).
           getQuantity();
71         if(length1 < length2){
72             length = length2;
73         } else if(length1 > length2){
74             length = length1;
75         }
76     }
77
78     return intlen(length);
79 }
80
81 size_t Warehouse::searchHighestPrice() {
82     size_t length = 0;
83     for (size_t i = 0; i < mArticles.size() - 1; i
           ++){
84         size_t length1 = mArticles.at(i).getPrice
           ();
85         size_t length2 = mArticles.at(i+1).
           getPrice();
86         if(length1 < length2){
87             length = length2;
88         } else if(length1 > length2){
89             length = length1;
90         }
91     }
92
93     return intlen(length);
94 }
95
96
97 bool isTbReal(scanner &scan){
98     return scan.symbol_is_real();
99 }
100
101 bool isTbString(scanner &scan){
102     return scan.symbol_is_string();
103 }
104
105 bool isTbInt(scanner &scan){
106     return scan.symbol_is_integer();
107 }
108
109 int parseArticleNumberOrQuantity(scanner &scan){
110     if(isTbInt(scan)){
111         return scan.get_integer();
112     }

```

```

113     throw std::exception("Unknown format");
114 }
115
116 string parseArticleName(scanner &scan){
117     if(isTbString(scan)){
118         return scan.get_string();
119     }
120     throw std::exception("Unknown format");
121 }
122
123 double parsePrice(scanner &scan){
124     if(isTbReal(scan)){
125         return scan.get_real();
126     }
127     throw std::exception("Unknown format");
128 }
129
130 Warehouse::Warehouse(std::string wareHouseName) :
131     mWareHouseName(wareHouseName), mArticles() {
132 }
133 Warehouse::~~Warehouse() {
134 }
135
136 void Warehouse::addArticle(Article const &
137     newArticle){
138     mArticles.push_back(newArticle);
139 }
140
141 size_t Warehouse::getNumberOfArticles() {
142     return mArticles.size();
143 }
144
145 void Warehouse::readArticlesFromFile(std::ifstream
146     &file){
147     scanner scan (file);
148
149     while(!scan.symbol_is_eof()){
150         if(isTbInt(scan)){
151             int articleNum =
152                 parseArticleNumberOrQuantity(scan);
153
154             scan.next_symbol();
155             scan.next_symbol();
156             if(isTbString(scan)){
157                 string articleName =
158                     parseArticleName(scan);
159                 scan.next_symbol();
160                 scan.next_symbol();

```

```

157         if (isTbInt(scan)){
158             size_t quant =
                parseArticleNumberOrQuantity
                (scan);
159             scan.next_symbol();
160             scan.next_symbol();
161             if (isTbReal(scan)){
162                 double price = parsePrice(
                    scan);
163                 scan.next_symbol();
164                 scan.next_symbol();
165                 Article art (articleNum,
                    articleName, quant,
                    price);
166                 mArticles.push_back(art);
167             }
168         }
169     }
170     } else{
171         throw std::exception("Unknown format")
            ;
172     }
173 }
174 }
175
176 void WareHouse::printArticleList( std::ostream &os
    ) {
177
178     sort(mArticles.begin(), mArticles.end(), comp
        ());
179
180     size_t colSpacing = 2;
181     size_t prec = 2;
182     size_t articleNumLength =
        searchHighestArticleNum() + colSpacing;
183     size_t articleNameLength = searchLongestName()
        + colSpacing;
184     size_t articleQuantityLength =
        searchHighestQuantity() + colSpacing;
185     size_t articlePriceLength = searchHighestPrice
        () + colSpacing + prec;
186
187
188     os << "Article list of Warehouse " <<
        mWareHouseName << endl;
189
190     for (std::vector<Article>::iterator it=
        mArticles.begin(); it!=mArticles.end(); ++
        it){

```

```

191         os << setw(articleNumLength) << left << it
           ->getArticleNumber();
192         os << setw(articleNameLength) << it->
           getArticleName();
193         os << setw(articleQuantityLength) << it->
           getQuantity();
194         os << right << setiosflags(ios::fixed) <<
           setw(articlePriceLength) <<
           setprecision(prec) << it->getPrice() <<
           endl;
195     }
196
197 }

```

Listing B.5: Testtreiber

```

1  //////////////////////////////////////
2  // Workfile      : Main.cpp
3  // Author       : Matthias Schett
4  // Date        : 20-04-2013
5  // Description  : Ware house management
6  // Remarks     : -
7  // Revision    : 0
8  //////////////////////////////////////
9  #include <vld.h>
10 #include "Article.h"
11 #include "WareHouse.h"
12 #include <fstream>
13 #include <iostream>
14 #include <string>
15
16 using namespace std;
17
18 void printTestHeader(int testNumber, ostream &
   stream){
19     stream << endl << endl;
20     if(testNumber < 10){
21         stream << "Testfall 0" << testNumber;
22     } else {
23         stream << "Testfall " << testNumber;
24     }
25     stream << endl << "—————" << endl <<
       endl;
26
27 }
28
29
30 int main(){
31     ofstream oFile("OutputA2.txt");
32     try{

```

```
33     printTestHeader(1, oFile);
34
35     Warehouse newWarehouse (string("
36         TestWarehouse"));
37
38     ifstream file ("Input.txt");
39
40     newWarehouse.readArticlesFromFile(file);
41
42     newWarehouse.printArticleList(oFile);
43
44     printTestHeader(2, oFile);
45
46     Warehouse newWarehouse2 (string("
47         TestWarehouse"));
48
49     ifstream file2 ("InputWrong.txt");
50
51     newWarehouse2.readArticlesFromFile(file2);
52
53     newWarehouse2.printArticleList(oFile);
54
55     file.close();
56     file2.close();
57 } catch(exception e){
58     oFile << e.what();
59 }
60
61 oFile.close();
62
63 cin.get();
64 return o;
65 }
```