```cpp
// KlasseZeit.cpp : Definiert den Einstiegspunkt für die Konsolenanwendung.
//
#include "stdafx.h"
#include <iostream>
#include <string>

// die nächsten beiden includes sind für die "schöne" Ausgabe und Bereitstellung als String
#include <iomanip>
#include <sstream>

using namespace std;

// Zuerst die Klassendeklaration, bei den Methoden nur Prototypen:
class Zeit {
private:
    int Stunde;
    int Minute;
    int Sekunde;
public:
    // die beiden Konstruktoren (als überladene Funktion):
    Zeit();
    Zeit(int, int, int);

    // die anderen Methoden:
    void setTime(int, int, int);
    string getTime();
    bool vorstellen(int, int, int);

    // ein Destruktor war nicht verlangt, brauchen wir auch nicht, der sähe so aus:
    // ~Zeit();
};

// die Implementierung der einzelnen Methoden zur Klasse "Zeit"
// außerhalb der Klassendeklaration mit Klasse::

// 1. Der Standardkonstruktor ohne Übergabeparameter setzt eine neue Uhr auf 00:00:00
Zeit::Zeit() {
    Stunde = 0;
    Minute = 0;
    Sekunde = 0;
}

// 2. Der zusätzliche Konstruktor ezugt eine Uhr mit übergebenen Zeitangaben:
Zeit::Zeit(int hh, int mm, int ss) {
    Stunde = hh;
    Minute = mm;
    Sekunde = ss;
}

void Zeit::setTime(int hh, int mm, int ss) {
    Stunde = hh;
    Minute = mm;
    Sekunde = ss;
}
```

```
55
56  string Zeit::getTime() {
57      // Dieser Trick wird im Unterricht bekannt gegeben (nicht examensrelevant, aber
        interessant...)
58      // dafür benötigt <iomanip> (für setfill/setw) und <sstream> (stringstream)
59      ostringstream temp;
60      temp << setfill('0') << setw(2) << Stunde << ":" << setw(2) << Minute << ":" <<
        setw(2) << Sekunde;
61      return temp.str();
62  }
63
64  bool Zeit::vorstellen(int hh, int mm, int ss) {
65      bool mussDatumkorrigieren = false;  // return-Wert vorbereiten
66      // Sekunden addieren:
67      Sekunde += ss;
68      //Korrektur bei Sekundenüberlauf:
69      Minute += (Sekunde/60);
70      Sekunde %= 60;
71      // Minuten addieren:
72      Minute += mm;
73      // Korrektur bei Minutenüberlauf:
74      Stunde += (Minute/60);
75      Minute %= 60;
76      // Stunden addieren:
77      Stunde += hh;
78      // Korrektur im Datum erforderlich?
79      if (Stunde>=24)
80          mussDatumkorrigieren = true;
81      Stunde %= 24;   // Stundendarstellung
82      return mussDatumkorrigieren;
83  }
84
85  // Hier jetzt das Szenario im Hauptprogramm:
86  int _tmain(int argc, _TCHAR* argv[])
87  {
88      Zeit Alarmzeit1;
89      Zeit* Alarmzeit2 = new Zeit(12,30,45);
90
91      cout << "Alarmzeit 1: " << Alarmzeit1.getTime() << endl;
92      cout << "Alarmzeit 2: " << Alarmzeit2->getTime() << endl << endl;
93
94      Alarmzeit1.setTime(15,23,10);
95      Alarmzeit2->setTime(8,8,8);
96
97      cout << "Alarmzeit 1: " << Alarmzeit1.getTime() << endl;
98      cout << "Alarmzeit 2: " << Alarmzeit2->getTime() << endl << endl;
99
100     cout << "Alarmzeit 1 vorstellen (10,10,10): ";
101     if (Alarmzeit1.vorstellen(10,10,10))
102         cout << "Bitte stellen Sie das Datum neu! ";
103     else
104         cout << "Das Datum bleibt korrekt! ";
105     cout << Alarmzeit1.getTime() << endl;
106
107     cout << "Alarmzeit 2 vorstellen (10,10,10): ";
108     if (Alarmzeit2->vorstellen(10,10,10))
```

```
109             cout << "Bitte stellen Sie das Datum neu! ";
110         else
111             cout << "Das Datum bleibt korrekt! ";
112         cout << Alarmzeit2->getTime() << endl;
113
114         cout << endl;
115         system("pause");
116         return 0;
117     }
118
119     /* Konsolenausgabe:
120     Alarmzeit 1: 00:00:00
121     Alarmzeit 2: 12:30:45
122
123     Alarmzeit 1: 15:23:10
124     Alarmzeit 2: 08:08:08
125
126     Alarmzeit 1 vorstellen (10,10,10): Bitte stellen Sie das Datum neu! 01:33:20
127     Alarmzeit 2 vorstellen (10,10,10): Das Datum bleibt korrekt! 18:18:18
128
129     Drücken Sie eine beliebige Taste . . .
130     */
```

```cpp
// Zeit2.cpp : Definiert den Einstiegspunkt für die Konsolenanwendung.
//
#include "stdafx.h"
#include <iostream>
#include <string>

// die nächsten beiden includes sind für die "schöne" Ausgabe und Bereitstellung als String
#include <iomanip>
#include <sstream>

using namespace std;

// Zuerst die Klassendeklaration, bei den Methoden nur Prototypen:
class Zeit {
private:
    int Stunde;
    int Minute;
    int Sekunde;
    // Neu in Aufgabe 2:
    bool Format; // per Vereinbarung: true=24 / false = AM/PM
public:
    // die beiden Konstruktoren (als überladene Funktion):
    Zeit();
    Zeit(int, int, int);

    // die anderen Methoden:
    void setTime(int=0, int=0, int=0); // erweitert um die Defaultwerte für "vergessene" Parameter
    string getTime();
    bool vorstellen(int=0, int=0, int=0); // erweitert um die Defaultwerte für "vergessene" Parameter

    // Neu in Aufgabe 2:
   int stellen(int=0, int=0, int=0); // erweitert um die Defaultwerte für "vergessene" Parameter
   void setFormat(bool);
};

// die Implementierung der einzelnen Methoden zur Klasse "Zeit"
// außerhalb der Klassendeklaration mit Klasse::

// 1. Der Standardkonstruktor ohne Übergabeparameter setzt eine neue Uhr auf 00:00:00
Zeit::Zeit() {
    setTime(0, 0, 0);   // natürlich können wir auch schon vorhandene Funktionen nutzen!
    Format = true;      // die neue Eigenschaften setzen wir auch gleich!
}

// 2. Der zusätzliche Konstruktor ezugt eine Uhr mit übergebenen Zeitangaben:
Zeit::Zeit(int hh, int mm, int ss) {
    setTime(hh, mm, ss);    // siehe oben...
    Format = true;
}

```

```
51  void Zeit::setTime(int hh, int mm, int ss) {
52      Stunde = hh;
53      Minute = mm;
54      Sekunde = ss;
55  }
56
57  string Zeit::getTime() {
58      // Dieser Trick wird im Unterricht bekannt gegeben (nicht examensrelevant, aber
        interessant...)
59      // dafür benötigt <iomanip> (für setfill/setw) und <sstream> (stringstream)
60      // Hier jetzt noch die Wahl zwischen den Ausgabeformaten zusätzlich
61      ostringstream temp;
62      if (Format)
63          temp << setfill('0') << setw(2) << Stunde << ":" << setw(2) << Minute << ":"
             << setw(2) << Sekunde;
64      else
65          temp << setfill('0') << setw(2) << Stunde%12 << ":" << setw(2) << Minute <<
            ":" << setw(2) << Sekunde << ((Stunde>12)?" PM":" AM");
66
67      return temp.str();
68  }
69
70  // neu in Aufgabe 2:
71  void Zeit::setFormat(bool Format) {
72      this->Format = Format; // nur damit wir auch den this-Zeiger mal einsetzen...
73  }
74
75  // auch neu in Aufgabe 2:
76  int Zeit::stellen(int hh, int mm, int ss) {
77      // dies ist eigentlich die alte "vorstellen(...)-Methode",
78      // jetzt auch für negative Werte brauchbar und mit Datumskorrektur in Anzahl
        Tagen
79     Stunde += hh;
80     Minute += mm;
81     Sekunde += ss;
82     int DatumNachstellen = 0;
83
84     // jetzt noch Korrektur:
85     // die Modulo-Division geht natürlich auch für negative Zahlen
86     // bei negativen Werten muss man lediglich 1 zusätzliche Bereinigung vornehmen
87     Minute  += int(Sekunde/60);
88     Sekunde  = Sekunde % 60;
89     if (Sekunde<0) {
90        Sekunde += 60;
91        Minute --;
92     }
93
94     Stunde += int(Minute/60);
95     Minute = Minute % 60;
96     if (Minute<0) {
97        Minute += 60;
98        Stunde --;
99     }
100
101    DatumNachstellen = int(Stunde/24);
102    Stunde  = (Stunde % 24);
```

```
103     if (Stunde<0) {
104        Stunde += 24;
105        DatumNachstellen --;
106     }
107
108     return DatumNachstellen;
109  }
110
111  bool Zeit::vorstellen(int hh, int mm, int ss) {
112      // da wir ja jetzt eine universelle Methode zum Vor- UND Zurück-Stellen haben,
113      // benutzen wir diese und schreiben zum Vorstellen nur noch:
114      if (stellen(hh,mm,ss)!=0) return true;
115      return false;
116  }
117
118  // Hier jetzt das Szenario im Hauptprogramm:
119  int _tmain(int argc, _TCHAR* argv[])
120  {
121      int Datumskorrektur; // neu in Aufgabe 2
122
123      Zeit Alarmzeit1;
124      Zeit* Alarmzeit2 = new Zeit(12,30,45);
125
126      cout << "Alarmzeit 1: " << Alarmzeit1.getTime() << endl;
127      cout << "Alarmzeit 2: " << Alarmzeit2->getTime() << endl << endl;
128
129      Alarmzeit1.setTime(15,23,10);
130      Alarmzeit2->setTime(8,8,8);
131
132      cout << "Alarmzeit 1: " << Alarmzeit1.getTime() << endl;
133      cout << "Alarmzeit 2: " << Alarmzeit2->getTime() << endl << endl;
134
135      cout << "Alarmzeit 1 vorstellen (10,10,10): ";
136      if (Alarmzeit1.vorstellen(10,10,10))
137          cout << "Bitte stellen Sie das Datum neu! ";
138      else
139          cout << "Das Datum bleibt korrekt! ";
140      cout << Alarmzeit1.getTime() << endl;
141
142      cout << "Alarmzeit 2 vorstellen (10,10,10): ";
143      if (Alarmzeit2->vorstellen(10,10,10))
144          cout << "Bitte stellen Sie das Datum neu! ";
145      else
146          cout << "Das Datum bleibt korrekt! ";
147      cout << Alarmzeit2->getTime() << endl;
148
149      // erweitertes Szenario in Aufgabe 2:
150      cout << endl;
151      Alarmzeit1.setTime(5);
152      cout << "Alarmzeit1: " << Alarmzeit1.getTime() << endl;
153
154      Datumskorrektur = Alarmzeit1.stellen(-8,-4);
155      cout << "Alarmzeit1: " << Alarmzeit1.getTime() << endl;
156      if (Datumskorrektur!=0) cout << "Bitte korrigieren Sie auch das Datum um " <<
         Datumskorrektur << " Tage!\n";
157
```

```
158     delete Alarmzeit2; // der Vollständigkeit halber...
159
160     cout << endl;
161     system("pause");
162     return 0;
163 }
164
165 /* Konsolenausgabe:
166 Alarmzeit 1: 00:00:00
167 Alarmzeit 2: 12:30:45
168
169 Alarmzeit 1: 15:23:10
170 Alarmzeit 2: 08:08:08
171
172 Alarmzeit 1 vorstellen (10,10,10): Bitte stellen Sie das Datum neu! 01:33:20
173 Alarmzeit 2 vorstellen (10,10,10): Das Datum bleibt korrekt! 18:18:18
174
175 Alarmzeit1: 05:00:00
176 Alarmzeit1: 20:56:00
177 Bitte korrigieren Sie auch das Datum um -1 Tage!
178
179 Drücken Sie eine beliebige Taste . . .
180 */
181
```

```
1  // Klasse Zeit Aufgabe 3 - Die große Lösung...
2  #include "stdafx.h"
3  #include <iostream>
4  #include <string>
5
6  // die nächsten beiden includes sind für die "schöne" Ausgabe und Bereitstellung als
   String
7  #include <iomanip>
8  #include <sstream>
9
10 using namespace std;
11
12 // Deklaration der Klasse Zeit mit Prototypen der Eigenschaften und Methoden:
13 class Zeit {
14 private:
15    int Stunde;
16    int Minute;
17    int Sekunde;
18    bool Format; // per Vereinbarung: true=24 / false = AM/PM
19 public:
20    Zeit();
21    Zeit(int, int, int);
22    void setTime(int, int, int);
23    bool vorstellen(int, int, int);
24    bool zurueckstellen(int, int, int);
25    string getTime();
26    int stellen(int, int, int);
27    void setFormat(bool);
28    string vergleiche(bool, int, int, int);
29    int vergleiche(int, int, int);
30 private:
31    int normalize(int&, int&, int&); // die Korrekturfunktion für Zeitüberläufe
   brauchen wir mehrmals...
32    string strZeit(int, int, int);   // die formatierten Zeitangaben brauchen wir
   auch öfter...
33 };
34
35 // Implementierung der Methoden: ----------------------------------------
36 Zeit::Zeit() { // der eigene Standard-Konstruktor
37    // wir nutzen die setTime-Methode, die wir sowieso brauchen:
38    setTime(0,0,0);
39    Format = true;
40 }
41 Zeit::Zeit(int hh, int mm, int ss) { // der Konstruktor mit Wertübergabe
42    // wir nutzen die setTime-Methode, die wir sowieso brauchen:
43    setTime(hh,mm,ss);
44    Format = true;
45 }
46 void Zeit::setTime(int hh=0, int mm=0, int ss=0) {
47    Stunde = hh;
48    Minute = mm;
49    Sekunde = ss;
50 }
51 bool Zeit::vorstellen(int hh=0, int mm=0, int ss=0) {
52      if (stellen(hh,mm,ss)!=0) return true;
53      return false;
```

```
54  }
55  bool Zeit::zurueckstellen(int hh=0, int mm=0, int ss=0) {
56       if (stellen(-hh,-mm,-ss)!=0) return true;
57       return false;
58  }
59  string Zeit::getTime() {
60     return strZeit(Stunde,Minute,Sekunde);
61  }
62  void Zeit::setFormat(bool Format) {
63       this->Format = Format; // nur damit wir auch den this-Zeiger mal brauchen...
64  }
65  int Zeit::stellen(int hh=0, int mm=0, int ss=0) {
66      // dies ist eigentlich die alte "vorstellen(...)-Methode",
67      // jetzt auch für negative Werte brauchbar und mit Datumskorrektur in Anzahl
        Tagen
68     Stunde += hh;
69     Minute += mm;
70     Sekunde += ss;
71     return normalize(Stunde, Minute, Sekunde);
72  }
73  string Zeit::vergleiche(bool Anzeige, int hh=0, int mm=0, int ss=0) {
74     // bool vorneweg, sonst funktioniert die Kombination aus Überladen und
       Standardwerten nicht (rechts nach links!)
75     ostringstream temp;
76     temp << "Die angegebene Zeit (" << strZeit(hh,mm,ss) << ") liegt um ";
77     int Differenz = vergleiche(hh, mm, ss);
78     bool Zukunft = (Differenz>=0);
79     hh = 0;
80     mm = 0;
81     ss = abs(Differenz);
82     Differenz = normalize(hh,mm,ss);
83     // wir wollen keine AM/PM-Anzeige, deshalb Format temporär auf 24-Stunden
       einstellen
84     bool tempFormat = Format;
85     Format = true;
86     temp << strZeit(hh,mm,ss) << " in der " << ((Zukunft)?"Zukunft":"Vergangenheit")
       << endl;
87     Format = tempFormat;
88     return temp.str();
89  }
90  int Zeit::vergleiche(int hh=0, int mm=0, int ss=0) {
91      int Differenz = (3600*hh + 60*mm + ss) - (3600*Stunde + 60*Minute + Sekunde);
92      return Differenz;
93  }
94  string Zeit::strZeit(int hh=0, int mm=0, int ss=0) {
95     // http://fara.cs.uni-potsdam.de/~kaufmann/?page=GenCppFaqs&faq=IntToString#Answ
96     // Dieser Trick wird im Unterricht bekannt gegeben (nicht examensrelevant, aber
       interessant...)
97     // Hier jetzt noch die Wahl zwischen den Ausgabeformaten zusätzlich
98     ostringstream temp;
99     if (Format)
100       temp << setfill('0') << setw(2) << hh << ":" << setw(2) << mm << ":" << setw
          (2) << ss;
101    else
102        temp << setfill('0') << setw(2) << hh%12 << ":" << setw(2) << mm << ":" <<
           setw(2) << mm << ((hh>12)?" PM":" AM");
```

```
103
104     return temp.str();
105  }
106  int Zeit::normalize(int& Stunde, int& Minute, int& Sekunde) {
107      // dies ist die Korrekturfunktion der bisherigen "stellen(...)-Methode":
108      // da die Zeitwerte in der Funktion geändert werden, aber auch außerhalb der
         Funktion gelten sollen,
109      // muss man die Werte als Zeiger übergeben.
110     int DatumNachstellen = 0;
111
112     // jetzt noch Korrektur:
113     // die Modulo-Division geht natürlich auch für negative Zahlen
114     // bei negativen Werten muss man lediglich 1 zusätzliche Bereinigung vornehmen
115     Minute  += int(Sekunde/60);
116     Sekunde  = Sekunde % 60;
117     if (Sekunde<0) {
118        Sekunde += 60;
119        Minute --;
120     }
121
122     Stunde += int(Minute/60);
123     Minute = Minute % 60;
124     if (Minute<0) {
125        Minute += 60;
126        Stunde --;
127     }
128
129     DatumNachstellen = int(Stunde/24);
130     Stunde  = (Stunde % 24);
131     if (Stunde<0) {
132        Stunde += 24;
133        DatumNachstellen --;
134     }
135
136     return DatumNachstellen;
137  }
138  /* ----------------------- Hauptprogramm ------------------------------ */
139  int main() {
140      int Datumskorrektur;
141
142      Zeit Alarmzeit1;
143      cout << "Alarmzeit1: " << Alarmzeit1.getTime() << endl;
144      Alarmzeit1.setFormat(false);
145      cout << "Alarmzeit1: " << Alarmzeit1.getTime() << endl;
146
147      Alarmzeit1.setTime(9,9,9);
148      cout << "Alarmzeit1: " << Alarmzeit1.getTime() << endl;
149
150      Zeit* Alarmzeit2 = new Zeit(12,12,12);
151      cout << "Alarmzeit2: " << Alarmzeit2->getTime() << endl;
152
153      if (Alarmzeit1.vorstellen(12,55,55))
154          cout << "Bitte stellen Sie auch das Datum! ";
155      else
156          cout << "Datum ist weiterhin korrekt! ";
157      cout << "Alarmzeit1: " << Alarmzeit1.getTime() << endl;
```

```
158     Alarmzeit1.setFormat(true);
159     cout << "Alarmzeit1: " << Alarmzeit1.getTime() << endl;
160
161     if (Alarmzeit2->vorstellen(12,55,55))
162         cout << "Bitte stellen Sie auch das Datum! ";
163     else
164         cout << "Datum ist weiterhin korrekt! ";
165     cout << "Alarmzeit2: " << Alarmzeit2->getTime() << endl;
166
167     Alarmzeit1.setTime(5);
168     cout << "Alarmzeit1: " << Alarmzeit1.getTime() << endl;
169
170     Datumskorrektur = Alarmzeit1.stellen(-8,-4);
171     cout << "Alarmzeit1: " << Alarmzeit1.getTime() << endl;
172     if (Datumskorrektur!=0) cout << "Bitte korrigieren Sie auch das Datum um " <<
        Datumskorrektur << " Tage!\n";
173
174     cout << "Jetzt vergleichen wir:\nZum Vergleich: Alarmzeit2 ist " << Alarmzeit2-
        >getTime() << endl;
175     cout << "Die Differenz zu 10:10:10 betraegt: " << Alarmzeit2->vergleiche
        (10,10,10) << " Sekunden\n";
176     cout << Alarmzeit2->vergleiche(true,10,10,10) << endl;
177     cout << "Die Differenz zu 00:08:05 betraegt: " << Alarmzeit2->vergleiche(0,8,5)
        << " Sekunden\n";
178     cout << Alarmzeit2->vergleiche(true,0,8,5) << endl;
179
180     delete Alarmzeit2; // der Vollständigkeit halber...
181
182     system("pause");
183     return 0;
184 }
185
186 /* Konsolenausgabe:
187 Alarmzeit1: 00:00:00
188 Alarmzeit1: 00:00:00 AM
189 Alarmzeit1: 09:09:09 AM
190 Alarmzeit2: 12:12:12
191 Datum ist weiterhin korrekt! Alarmzeit1: 10:05:05 PM
192 Alarmzeit1: 22:05:04
193 Bitte stellen Sie auch das Datum! Alarmzeit2: 01:08:07
194 Alarmzeit1: 05:00:00
195 Alarmzeit1: 20:56:00
196 Bitte korrigieren Sie auch das Datum um -1 Tage!
197 Jetzt vergleichen wir:
198 Zum Vergleich: Alarmzeit2 ist 01:08:07
199 Die Differenz zu 10:10:10 betraegt: 32523 Sekunden
200 Die angegebene Zeit (10:10:10) liegt um 09:02:03 in der Zukunft
201
202 Die Differenz zu 00:08:05 betraegt: -3602 Sekunden
203 Die angegebene Zeit (00:08:05) liegt um 01:00:02 in der Vergangenheit
204
205 Drücken Sie eine beliebige Taste . . .
206 */
```