Karlsruher Institut für Technologie
Institut für Technik der Informationsverarbeitung

Aufgaben zum C++ Tutorium

Dipl.-Ing. Adnene Gharbi, Dipl.-Ing. Christoph Roth,
Dipl.-Ing.(FH) Tobias Schwalb, Dipl.-Ing. Michael Tansella
cand. B.Sc. Andreas Kleff, cand. B.Sc. Felix Mauch

Institutsleitung
Prof. Dr.-Ing. K. D. Müller-Glaser
Prof. Dr.-Ing. J. Becker
Prof. Dr. rer. nat. W. Stork

KIT - Universität des Landes Baden Württemberg und nationales
Forschungszentrum in der Helmholtz-Gemeinschaft

Vorwort

Diese Aufgaben sollen Ihnen helfen, Ihre erworbenen Kenntnisse aus Vorlesung und Übung während dem C++ Tutorium am Rechenzentrum praktisch umzusetzen.

Zu jeder Aufgabe ist angegeben, welche Vorkenntnisse Sie brauchen, um diese Aufgabe lösen zu können. Lesen Sie sich also die entsprechenden Abschnitte im Kompendium und in den Richtlinien durch, bevor Sie mit der Aufgabe beginnen.

Vorwissen:
Kompendium: Kapitel X.Y
Richtlinien: Kapitel Z

Zu Ihrer Hilfe sind während dem Tutorium auch Tutoren anwesend. Also zögern Sie nicht, diese um Hilfe zu bitten, wenn Ihnen das Kompendium oder weitere Literaturquellen nicht mehr weiterhelfen. Erwarten Sie allerdings keine Komplettlösungen der Tutoren.

Verbesserungsvorschläge, sowie jede Art von Kritik zu diesem Dokument, sind sehr erwünscht. Bitte senden Sie diese an *michael.tansella@kit.edu* oder *tobias.schwalb@kit.edu*.

©ITIV, Karlsruhe 2011
3. Auflage

Aufgabe 1 Hello Students

Vorwissen:
Kompendium: Kapitel 1
Richtlinien: -

Schreiben Sie ein Programm, welches Ihren Namen, Adresse, Email-Adresse und Telefonnummer auf dem Bildschirm ausgibt. Fügen Sie dabei auch Kommentare zum Namen des Programmierers, Programmnamen, Erstellungsdatum und zur Version hinzu. Zusätzlich kommentieren Sie auch Ihren Code.

Beispielausgabe:

```
Max Musterstudent
Studentenallee 6
76131 Karlsruhe

max.musterstudent@student.kit.edu
0721123456
```

Aufgabe 2 Hello Variablen

Vorwissen:
Kompendium: Kapitel 2.1 - 2.5
Richtlinien: Kapitel 1.1

Ergänzen Sie nun das Programm aus Aufgabe 1, indem Sie die Daten zunächst in Variablen speichern und anschließend den Inhalt der Variablen ausgeben.

Aufgabe 3 Hello Eingabe

Vorwissen:
Kompendium: Kapitel 2.1 - 2.5
Richtlinien: Kapitel 1.1

Erweitern Sie Ihr Programm aus Aufgabe 2 nun dahingehend, dass Sie die Daten von der Tastatur einlesen und die Eingaben in Ihren vorhandenen Variablen speichern. Geben Sie anschließend wieder den Inhalt Ihrer Variablen aus.

Hinweis: Vermeiden Sie Leerzeichen in Ihren Eingaben.

Aufgabe 4 Operatoren

Vorwissen:
Kompendium: Kapitel 2.8, Anhang A
Richtlinien: Kapitel 1.4

Erstellen Sie ein Programm mit drei Ganzzahlen. Weisen Sie nun der ersten Zahl den Wert '5' zu. Versuchen Sie nun, den anderen Variablen nur mit Rechenschritten die gewünschten Werte zuzuweisen. Erzeugen Sie mit weiteren Rechenoperationen die folgende Bildschirmausgabe:

Hinweis: Für dieses Programm gibt es zahllose Möglichkeiten. Verwenden Sie viele verschiedene Rechenoperatoren. Sie können beispielsweise versuchen abgesehen von der ersten '5' keine Zahl in den Quellcode zu schreiben.

Beispielausgabe:

```
Erste Zahl: 5
Zweite Zahl: 10
Dritte Zahl: 15
--ein paar kleine Rechnungen--
Erste Zahl: 25
Zweite Zahl: -5
Dritte Zahl: 16
```

Aufgabe 5 Wo liegt die Zahl?

Vorwissen:
Kompendium: Kapitel 3.1 und Kapitel 3.2
Richtlinien: Kapitel 2

Schreiben Sie ein Programm, das die Größe einer eingegebenen Zahl schätzt. Das Programm soll unterscheiden können, ob die Zahl kleiner als 0 ist, genau 0 ist, zwischen 0 und 100 liegt, genau 100 ist oder größer als 100 ist. Liegt die Zahl zwischen 0 und 100, soll der Benutzer gefragt werden, ob die Zahl noch genauer eingeordnet werden soll. Falls die Zahl noch weiter eingeordnet werden soll, soll unterschieden werden, ob die Zahl kleiner als 50 oder größer gleich 50 ist.

Beispielausgabe:

```
Bitte eine ganze Zahl eingeben: 34
Die Zahl liegt zwischen 0 und 100.
Weiter eingrenzen? [j/n] j
Die Zahl ist kleiner als 50
```

Aufgabe 6 Wo liegt die Zahl genau?

Vorwissen:
Kompendium: Kapitel 3.4
Richtlinien: Kapitel 2

Erweitern Sie Ihr Programm aus Aufgabe 5. Liegt die Zahl zwischen 0 und 100, soll sie genau bestimmt werden. Überlegen Sie sich hierzu vorher einen geeigneten Algorithmus zur Bestimmung der Zahl.

Beispielausgabe:

```
Bitte eine ganze Zahl eingeben: 77
Die Zahl liegt zwischen 0 und 100.
Weiter eingrenzen? [j/n] j
Die Zahl ist groesser als 50
Die Zahl ist groesser als 75
Die Zahl ist kleiner als 87
Die Zahl ist kleiner als 81
Die Zahl ist kleiner als 78
Die Zahl ist groesser als 76
Die eingegebene Zahl ist gleich 77
```

Aufgabe 7 Mathematikrechner

Vorwissen:
Kompendium: Kapitel 3.1, 3.2 und 3.4
Richtlinien: Kapitel 2.1 - 2.4

Schreiben Sie einen kleinen Mathematikrechner der nacheinander die erste Zahl, das Rechenzeichen und dann die zweite Zahl einliest. Dabei sollen auch (negative) Gleitpunktzahlen zugelassen sein.

Anschließend soll die eingegebene Aufgabe vom Programm berechnet und das Ergebnis auf dem Bildschirm ausgegeben werden. Zusätzlich soll das Programm in einer Schleife laufen, sodass der Benutzer mehrere Aufgaben hintereinander rechnen kann, ohne das Programm immer wieder neu starten zu müssen. Eine Möglichkeit zum Beenden soll allerdings trotzdem gegeben sein (z.B. mit einer zusätzlichen Abfrage am Ende jeder Rechenaufgabe).

Beispielausgabe:

```
Geben Sie bitte die erste Gleitpunktzahl ein: 2.34
Geben Sie das Rechenzeichen ein: +
Geben Sie bitte die zweite Gleitpunktzahl ein: 1.11
Ergebnis: 2.34 + 1.11 = 3.45

Moechten Sie das Programm nochmal ausfuehren (j/n)? n
```

Aufgabe 8 Reihendarstellung von versch. Funktionen

Vorwissen:
Kompendium: Kapitel 3.3
Richtlinien: Kapitel 2.5

Erstellen Sie ein Programm, das die Reihendarstellung der e-Funktion, die des Sinus, die des Kosinus, sowie die geometrische Reihe ausgeben kann.
Geben Sie dem Benutzer zunächst ein Menü vor, in dem er zwischen den vier Reihen wählen kann. Danach soll der Benutzer gefragt werden, wie viele Elemente ausgegeben werden sollen.
Falls der Benutzer eine ungültige Wahl trifft, sollen Glieder der geometrischen Reihe ausgegeben werden.
Benutzen Sie für das Menü Switch-Case-Anweisungen. Eine Beispielausgabe finden Sie bei Aufgabe 9.
Hinweis:

$$e^x = \sum_{n=0}^{\infty} \frac{1}{n!} x^n \qquad \sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!} x^{2n+1} \qquad \cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} \qquad \frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$$

Aufgabe 9 Reihendarstellung von Funktionen mit Funktionen

Vorwissen:
Kompendium: Kapitel 4.1 - 4.5
Richtlinien: Kapitel 3.1 - 3.4

Ändern Sie das Programm der letzten Aufgabe ab, sodass Sie für die Ausgabe von Sinus, Kosinus, e-Funktion und geometrischer Reihe jeweils eine Funktion schreiben, die sich um die Einzelheiten kümmert. Innerhalb des Switch-Case Blocks in Ihrem Hauptprogramm müssen Sie dann nun nur noch die entsprechende Funktion aufrufen.

1. Schreiben Sie Ihre Funktionen zunächst so, dass immer die ersten fünf Reihenglieder ausgegeben werden. Implementieren Sie diese Methoden und testen Sie Ihr Programm.

2. Ändern Sie Ihre Funktionen nun so, dass Sie beim Aufruf der Funktion festlegen können, wie viele Reihenglieder ausgegeben werden können.

Beispielausgabe:

```
Welche Funktion soll ausgegeben werden?
(1) e-Funktion
(2) Sinusfunktion
(3) Kosinusfunktion
(4) Geometrische Reihe
Funktion: 3
Wie viele Reihenglieder sollen ausgegeben werden? 3
Es werden die ersten 3 Reihenglieder der Kosinusfunktion ausgegeben.
cos(x) = 1 - (1/(2!))*x^2 + (1/(4!))*x^4
```

Aufgabe 10 Fakultät berechnen, Fehler abfangen

Vorwissen:
Kompendium: Kapitel 4.1 - 4.5.1
Richtlinien: Kapitel 3

Schreiben Sie eine Funktion, welche die Fakultät einer Ganzzahl mit Hilfe einer Schleife berechnet und zurück gibt. Achten Sie dabei auch darauf, dass die Fakultät von Null gleich Eins ist.
Schreiben Sie nun ein Programm, welches

- eine Ganzzahl von der Tastatur einliest
- wenn diese negativ ist eine Fehlermeldung ausgibt
- ansonsten die eben geschriebene Funktion zur Fakultätsberechnung aufruft
- und das Ergebnis der Berechnung auf dem Bildschirm ausgibt

Hinweis: Für alle natürlichen Zahlen n ist $n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n$
Zusatz: Die Fakultätsfunktion lässt sich auch sehr effizient als rekursive Funktion implementieren.
Beispielausgabe:

```
Bitte eine Ganzzahl eingeben: 5
5! = 120
```

Aufgabe 11 Quadrieren mit Zeigerübergabe

Vorwissen:
Kompendium: Kapitel 4.5.3
Richtlinien: Kapitel 3.5

Erstellen Sie eine Funktion, welche eine als Zeiger übergebene Variable quadriert. Schreiben Sie anschließend ein kleines Testprogramm, welches eine Zahl von der Tastatur einliest und diese anschließend quadriert. Rufen Sie hierzu die von Ihnen erstellte Quadradfunktion auf.
Beispielausgabe:

```
Bitte geben Sie eine Zahl ein: 4
4^2 ergibt 16
```

Aufgabe 12 Quadrieren mit Übergabe als Referenz

Vorwissen:
Kompendium: Kapitel 4.5.2
Richtlinien: Kapitel 3.5

Schreiben Sie nun ihre Funktion aus Aufgabe 11 so um, dass ihr eine Referenz als Parameter übergeben werden kann. Rufen Sie in Ihrem Hauptprogramm die neue Funktion auf.

Aufgabe 13 Multiplikation von komplexen Zahlen

Vorwissen:
Kompendium: Kapitel 2.11, Kapitel 4
Richtlinien: Kapitel 2, 3

In dieser Aufgabe soll eine Funktion zur Multiplikation zweier komplexer Zahlen implementiert werden. Stellen Sie hierzu eine komplexe Zahl mit einem Array aus zwei Integer-Zahlen dar.

$$z = 2 + j3 \Rightarrow arr[2] = \{2, 3\}$$

Schreiben Sie nun eine Funktion, welche drei komplexe Zahlen als Parameter akzeptiert. Die ersten beiden Zahlen sollen miteinander multipliziert werden und das Ergebnis in der dritten gespeichert werden.
Erstellen Sie anschließend ein kleines Testprogramm, das zwei komplexe Zahlen vom Benutzer abfragt und diese mit der eben von Ihnen erstellten Funktion miteinander multipliziert.
Beispielausgabe:

```
Bitte geben Sie den Realteil der ersten Zahl ein: -1
Bitte geben Sie den Imaginaerteil der ersten Zahl ein: 3
Bitte geben Sie den Realteil der zweiten Zahl ein: 2
Bitte geben Sie den Imaginaerteil der zweiten Zahl ein: -1
zahl1 * zahl2 = 1 + j7
```

Aufgabe 14 Multiplikation von 3×3-Matrizen

Vorwissen:
Kompendium: Kapitel 2.11, Kapitel 4
Richtlinien: Kapitel 2, 3

Nachdem Sie in Aufgabe 13 die Multiplikation komplexer Zahlen umgesetzt haben, erstellen Sie nun eine Funktion zum Multiplizieren von 3×3-Matrizen nach dem Falkschen Schema.
Die Multiplikationsfunktion soll wieder drei Argumente erhalten: die beiden eingegebenen Matrizen, sowie die leere Matrix für das Ergebnis.
Tipp: Schreiben Sie für die Ein- und Ausgabe ebenfalls eigene Funktionen, so müssen Sie diese nur einmal schreiben und können Sie für beide Matrizen ausführen.
Beispielausgabe:

```
Matrix A eingeben:
Element a11 eingeben: 1
Element a12 eingeben: -2
Element a13 eingeben: 0
Element a21 eingeben: 2
Element a22 eingeben: -3
Element a23 eingeben: 0
Element a31 eingeben: 2
Element a32 eingeben: 1
Element a33 eingeben: -2
Eingegebene Matrix:
1 -2 0
2 -3 0
2 1 -2

Matrix B eingeben:
Element b11 eingeben: 0
Element b12 eingeben: 1
Element b13 eingeben: -2
Element b21 eingeben: 3
Element b22 eingeben: 3
Element b23 eingeben: -3
Element b31 eingeben: 1
Element b32 eingeben: -2
Element b33 eingeben: 4
Eingegebene Matrix:
0 1 -2
3 3 -3
1 -2 4

A * B:
-6 -5 4
-9 -7 5
1 9 -15
```

Aufgabe 15 Eine kleine Autoklasse

Vorwissen:
Kompendium: Kapitel 5.1 - 5.3
Richtlinien: Kapitel 6

Schreiben Sie eine Klasse für ein Auto. Das Auto soll folgende privaten Attribute besitzen:

- Marke
- Baujahr
- Beschleunigung von 0 auf 100 km/h

- Höchstgeschwindigkeit

Erstellen Sie außerdem die folgenden Methoden:

- Get- & Set-Methoden für sämtliche Attribute
- `beschleunigen`: Erstellen Sie die Ausgabe wie in der Beispielausgabe zu sehen und geben Sie nach dem Beschleunigungsvorgang die erreichte Höchstgeschwindigkeit aus.
- `hupen`: Erzeugen Sie eine Ausgabe für das Hupen.

Schreiben Sie anschließend ein kleines Testprogramm für Ihre Autoklasse, welches ein Auto Objekt erzeugt, dieses beschleunigt und anschließend hupt.
Verwenden Sie für die Automarke einen eigenen Datentyp, der mindestens die folgenden Werte annehmen kann: *Mercedes, Audi, Volkswagen, BMW.*
Zusatz: Ihr Auto soll Bescheid sagen, wenn es erzeugt wird und wenn es zerstört wird.
Beispielausgabe:

```
Ein neues Auto wird erstellt!
Beschleunige bis zur Maximalgeschwindigkeit... ... ... ... ... ... ... ... ... ...
... ... ... ... ... ... ... ... ... ... ...
183 km/h erreicht.
Tuutuuuuut!
Ein Auto wird zerstoert
```

Aufgabe 16 String-Behandlung

Vorwissen:
Kompendium: Kapitel 5.4
Richtlinien: Kapitel 6

Erzeugen Sie einen String mit dem Inhalt "*Ich studiere gerne an der Universitaet Karlsruhe.*".

1. Manipulieren Sie den String nun, sodass er nach der Manipulation lautet: "*Ich studiere Elektro- und Informaionstechnik am Karlsruher Institut fuer Technologie. :)*"
2. Fügen Sie nun noch in den Satz ein, wie lange Sie bereits am KIT studieren.
3. Entfernen Sie jetzt wieder das Smiley am Ende des Satzes.

Beispielausgabe:

```
Ich studiere gerne an der Universitaet Karlsruhe.
Etwas Manipulation...
Ich studiere Elektro- und Informationstechnik am Karlsruher Institut fuer
Technologie. :)
Nochmal etwas Manipulation...
Ich studiere seit 2 Semestern Elektro- und Informationstechnik am Karlsruher Institut
fuer Technologie. :)
Das Smiley wirkt etwas unprofessionell...
Ich studiere seit 2 Semestern Elektro- und Informationstechnik am Karlsruher Institut
fuer Technologie.
```

Aufgabe 17 Ein paar Fahrzeuge

Vorwissen:
Kompendium: Kapitel 6, 7, 8
Richtlinien: Kapitel 6

Erzeugen Sie zunächst eine Fahrzeugsklasse nach dem Beispiel der Autoklasse in Aufgabe 13. Erweitern Sie die Klasse um die Methode `steckbrief()`, welche Marke, Baujahr, Beschleunigung, sowie Höchstgeschwindigkeit des Fahrzeugs ausgibt.
Leiten Sie nun von dieser Basisklasse die Klassen LKW, Auto und Motorrad ab.
Die abgeleiteten Klassen sollen sich in der Ausgabe der Methode `hupen()` unterscheiden:

Fahrzeug "Ich hupe!"

Auto "Tuutuut!"

LKW "Troeoeoeoet!"

Motorrad "Tuetuet!"

Außerdem soll im Steckbrief ausgegeben werden, um welchen Fahrzeugtyp es sich handelt.
Erstellen Sie nun ein Testprogramm, welches ein dynamisches Objekt der Klasse `Fahrzeug` erstellt. Bieten Sie dem Benutzer hierzu zuerst ein Menü, um den Fahrzeugtyp zu wählen. Geben Sie anschließend die Marke ebenfalls über ein Benutzermenü ein. Lesen Sie nun noch die restlichen Fahrzeugdaten ein.
Wenn der Benutzer einen ungültigen Fahrzeugtyp eingibt, soll ein Fahrzeug vom Typ Fahrzeug erstellt werden.
Wird eine ungültige Marke eingegeben, setzen Sie die Marke auf "unbekannt".

Geben Sie nun den Steckbrief des Fahrzeugs aus und hupen Sie ein mal.
Wenn ein Fahrzeug erstellt wird, soll eine entsprechende Meldung über die Erzeugung informieren und den Fahrzeugtyp nennen. Das selbe soll beim Zerstören eines Fahrzeugs geschehen.
*Hinweis: Verwenden Sie für jede Klasse getrennte Header- und *.cpp-Dateien.*
Beispielausgabe:

```
Was fuer ein Fahrzeug soll erstellt werden?
(1) Auto
(2) LKW
(3) Motorrad
Eingabe: 3
Ein neues Fahrzeug wird erstellt!
Ein neues Motorrad wird erstellt!
Marke eingeben:
(1) Mercedes
(2) Audi
(3) Volkswagen
(4) BMW
(5) Andere
Eingabe: 4
Baujahr eingeben: 1987
Hoechstgeschwindigkeit eingeben: 190
Beschleunigung von 0 auf 100 km/h in Sekunden eingeben: 5.4

Steckbrief:
Ich bin ein Motorrad der Marke BMW
Mein Baujahr ist 1987
Ich komme in 5.4 Sekunden von 0 auf 100 km/h
Meine Hoechstgeschwindigkeit ist 190 km/h

Beschleunige bis zur Maximalgeschwindigkeit... ... ... ... ... ... ... ... ... ...
... ... ... ... ... ... ... ... ... ...
190 km/h erreicht.
Tuetuet!

Ein Motorrad wird zerstoert!
Ein Fahrzeug wird zerstoert!
```

Aufgabe 18 Garage mit fstream einlesen

Vorwissen:
Kompendium: Kapitel 8, Kapitel 9, Anhang B.1.1
Richtlinien: Kapitel 5, Kapitel 6

Jetzt wollen wir das Programm aus Aufgabe 15 erweitern. Erstellen Sie eine Klasse namens Garage, welche in einem Vektor als Attribut Fahrzeuge aufnehmen kann. Erstellen Sie außerdem eine Methode, welche eine Textdatei *garage.txt* einlesen kann und daraus die Informationen der geparkten Fahrzeuge sammelt. Ein Eintrag in der Textdatei ist nach dem folgenden Schema aufgebaut:

```
[Fahrzeugtyp]
Marke: automarke
Baujahr: jahreszahl
Geschwindigkeit: maxGeschwindigkeit
Beschleunigung: beschleunigung
```

Kommentarzeilen in der Textdatei beginnen mit einem #-Zeichen.
Erstellen Sie für jeden Eintrag in der Textdatei ein Objekt in Ihrem Fahrzeug-Vektor.
Schreiben Sie nun eine Methode der Klasse Garage, welche den Inhalt der Garage, also des Fahrzeugsvektors ausgibt. Hupen Sie nach jedem ausgegebenen Fahrzeug einmal mit demselben.
Überlegen Sie sich, wie ein Testprogramm für Ihre Garagenklasse aussehen müsste und erstellen Sie dieses.

Kapitel 1

Lösungen

Aufgabe 1 01.cpp

```cpp
// Erstellt: 16.02.2010
// Autor: Felix Mauch
// Uebungsaufgabe 1
// Beschreibung: Gibt eine Visitenkarte auf dem Bildschirm aus.

#include <iostream>
using namespace std;

// Dies ist die Hauptfunktion
int main() {
  cout << "Max Mustermann" << endl;
  cout << "Studentenallee 6" << endl;
  cout << "76131 Karlsruhe" << endl;
  cout << endl; // Leere Zeile
  cout << "max.mustermann@student.kit.edu" << endl;
  cout << "0721123456" << endl;

  return 0;
}
```

Aufgabe 2 02.cpp

```cpp
// Erstellt: 16.02.2010
// Autor: Felix Mauch
// Uebungsaufgabe 2
// Beschreibung: Gibt eine Visitenkarte auf dem Bildschirm aus. Diese Version speichert zuerst alle Daten in Variablen.

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

// Dies ist die Hauptfunktion
int main() {
  // Zuerst die Daten festlegen
  string vorname = "Max";
  string nachname = "Mustermann";
  string strasse = "Studentenallee";
  unsigned short hausnr = 6;
  unsigned int plz = 76131;
  string stadt = "Karlsruhe";
  string email = "max.mustermann@student.kit.edu";
  string telefon = "0721123456";

  // Daten ausgeben
  cout << vorname << " " << nachname << endl;
  cout << strasse << " " << hausnr << endl;
  cout << plz << " " << stadt << endl;
  cout << endl;
```

```
27   cout << email << endl;
28   cout << telefon << endl;
29
30   return 0;
31 }
```

Aufgabe 3 03.cpp

```
1  // Erstellt: 16.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 3
4  // Beschreibung: Gibt eine Visitenkarte auf dem Bildschirm aus. Diese ↩
       Version liest die Daten von der Eingabezeile ein und speichert ↩
       diese dann in Variablen.
5
6  #include <iostream>
7  #include <string>
8  using namespace std;
9
10 int main() {
11   // Variablen definieren
12   string nachname, vorname;
13   string strasse;
14   unsigned short hausnr;
15   unsigned int plz;
16   string stadt;
17   string email;
18   string telefon;
19
20   // Daten von der Kommandozeile einlesen
21   cout << "Bitte geben Sie Ihren Nachnamen ein: ";
22   cin >> nachname;
23   cout << "Bitte geben Sie Ihren Vornamen ein: ";
24   cin >> vorname;
25   cout << "Bitte geben Sie Ihre Strasse ein: ";
26   cin >> strasse;
27   cout << "Bitte geben Sie Ihre Hausnummer ein: ";
28   cin >> hausnr;
29   cout << "Bitte geben Sie Ihre Postleitzahl ein: ";
30   cin >> plz;
31   cout << "Bitte geben Sie Ihren Wohnort ein: ";
32   cin >> stadt;
33   cout << "Bitte geben Sie Ihre Emailadresse ein: ";
34   cin >> email;
35   cout << "Bitte geben Sie Ihre Telefonnummer ein: ";
36   cin >> telefon;
37   cout << endl << "Eingegebene Daten: " << endl;
38
39   // Daten ausgeben
40   cout << vorname << " " << nachname << endl;
41   cout << strasse << " " << hausnr << endl;
42   cout << plz << " " << stadt << endl;
43   cout << endl;
44   cout << email << endl;
45   cout << telefon << endl;
46
47   return 0;
48 }
```

Aufgabe 4 04.cpp

```cpp
// Erstellt: 16.02.2010
// Autor: Felix Mauch
// Uebungsaufgabe 4
// Beschreibung: Verwendung verschiedener Rechenoperatoren.

#include <iostream>
using namespace std;

int main() {
  int zahl1, zahl2, zahl3;

  zahl1 = 5; // Nur die erste Variable wird direkt definiert
  zahl2 = 2 * zahl1;
  zahl3 = zahl1 + zahl2;

  // Erste Ausgabe
  cout << "Erste Zahl: " << zahl1 << endl;
  cout << "Zweite Zahl: " << zahl2 << endl;
  cout << "Dritte Zahl: " << zahl3 << endl;

  cout << "-- ein paar kleine Rechnungen--" << endl;

  // Berechnungen
  zahl1 = zahl1 * 5; // entspricht zahl1 *= 5;
  zahl2 -= zahl3; // entspricht zahl2 = zahl2 - zahl3;
  zahl3++; // entspricht zahl3 += 1; oder zahl = zahl3 + 1;

  // Zweite Ausgabe
  cout << "Erste Zahl: " << zahl1 << endl;
  cout << "Zweite Zahl: " << zahl2 << endl;
  cout << "Dritte Zahl: " << zahl3 << endl;

  return 0;
}
```

Aufgabe 5 05.cpp

```cpp
// Erstellt: 16.02.2010
// Autor: Felix Mauch
// Uebungsaufgabe 5
// Beschreibung: Ordnet eine eingegebene Zahl in einen Zahlenraum ein.

#include <iostream>
using namespace std;

int main() {
  int zahl; // Einzugrenzende Zahl
  char abfrage; // Abfrage nach weiterer Eingrenzung

  cout << "Bitte eine ganze Zahl eingeben: ";
  cin >> zahl;

  if( zahl < 0 ) {
    // 1. Fall: Zahl ist kleiner als 0
    cout << "Die eingegebene Zahl ist kleiner als 0" << endl;
  } else if( zahl == 0 ) {
    // 2.Fall: Zahl ist gleich 0
```

```
21      cout << "Sie haben \'0\' eingegeben!" << endl;
22    } else if( zahl <= 100 ) {
23      // 3. Fall: Zahl ist zwischen 0 und 100
24      cout << "Die Zahl liegt zwischen 0 und 100." << endl;
25      cout << "Weiter eingrenzen? [j/n] ";
26      cin >> abfrage;
27      if( abfrage == 'j' ) {
28        if( zahl < 50) {
29          cout << "Die Zahl ist kleiner als 50" << endl;
30        } else {
31          cout << "Die Zahl ist groesser oder gleich 50" << endl;
32        }
33      } else if( abfrage == 'n' ) {
34        cout << "Es folgt keine weitere Einordnung!" << endl;
35      } else {
36        cout << "Ungueltige Eingabe! Es folgt keine weitere Einordnung" ↩
             << endl;
37      }
38    } else {
39      // 4. Fall: Zahl ist größer als 100
40      cout << "Die Zahl ist groesser als 100" << endl;
41    }
42
43    return 0;
44  }
```

Aufgabe 6 06.cpp

```
1  // Erstellt: 16.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 6
4  // Beschreibung: Ordnet eine eingegebene Zahl in einen Zahlenraum ein.↩
       Liegt sie zwischen 0 und 100, wird sie genau 'erraten'.
5
6  #include <iostream>
7  using namespace std;
8
9  int main() {
10   int zahl;
11   char abfrage;
12
13   cout << "Bitte eine ganze Zahl eingeben: ";
14   cin >> zahl;
15
16   if( zahl < 0 ) {
17     cout << "Die eingegebene Zahl ist kleiner als 0" << endl;
18   } else if( zahl == 0 ) {
19     cout << "Sie haben \'0\' eingegeben!" << endl;
20   } else if( zahl < 100 ) {
21     cout << "Die Zahl liegt zwischen 0 und 100." << endl;
22     cout << "Weiter eingrenzen? [j/n] ";
23     cin >> abfrage;
24     if( abfrage == 'j' ) {
25       int unten = 0;
26       int oben = 100;
27       bool erraten = false;
28       while( !erraten ) {
29         int grenze = unten + ( oben - unten ) / 2; // Intervall teilen
30         if( zahl < grenze ) { // untere IntervallhÃ¤lfte
```

```
31            cout << "Die Zahl ist kleiner als " << grenze << endl;
32            oben = grenze;
33          } else if( zahl > grenze ) { // obere IntervallhÃ¤lfte
34            cout << "Die Zahl ist groesser als " << grenze << endl;
35            unten = grenze;
36          } else {
37            erraten = true; // Schleifenabbruch
38            cout << "Die eingegebene Zahl ist gleich " << grenze << endl;
39          }
40        }
41
42      } else if( abfrage == 'n' ) {
43        cout << "Es folgt keine weitere Einordnung!" << endl;
44      } else { // Abfangen einer Fehleingabe
45        cout << "Ungueltige Eingabe! Es folgt keine weitere Einordnung"
              << endl;
46      }
47    } else if( zahl == 100 ) {
48      cout << "Sie haben \'100\' eingegeben!" << endl;
49    } else {
50      cout << "Die Zahl ist groesser als 100" << endl;
51    }
52
53    return 0;
54  }
```

Aufgabe 7 07.cpp

```
1  // Erstellt: 28.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 7
4  // Beschreibung: Ein einfacher kleiner Rechner, der zwei eingegebene
       Zahlen addieren, subtrahieren, multiplizieren und dividieren kann.
5
6  #include <iostream>
7  using namespace std;
8
9  int main() {
10   char repeat = 'j';
11
12   while( repeat == 'j' || repeat == 'J' ) { // Wiederholung des
         Programms
13     float zahl1, zahl2, ergebnis;
14     char rechenzeichen;
15
16     // Eingabe
17     cout << "Geben Sie bitte die erste Gleitpunktzahl ein: " ;
18     cin >> zahl1;
19     cout << "Geben Sie das Rechenzeichen ein: ";
20     cin >> rechenzeichen;
21     cout << "Geben Sie bitte die zweite Gleitpunktzahl ein: " ;
22     cin >> zahl2;
23
24     // Rechenart bestimmen
25     if( rechenzeichen == '+' ) {
26       ergebnis = zahl1 + zahl2;
27     } else if ( rechenzeichen == '-' ) {
28       ergebnis = zahl1 - zahl2;
```

```
29      } else if ( rechenzeichen == '*' ) {
30        ergebnis = zahl1 * zahl2;
31      } else if ( rechenzeichen == '/' ) {
32        ergebnis = zahl1 / zahl2;
33      }
34
35      // Ausgabe
36      cout << "Ergebnis: " << zahl1 << " " << rechenzeichen << " " << ↩
            zahl2 << " = " << ergebnis << endl;
37
38      cout << endl << "Moechten Sie das Programm nochmal ausfuehren (j/n↩
            )? ";
39      cin >> repeat;
40      cout << endl;
41    }
42
43    return 0;
44  }
```

Aufgabe 8 08.cpp

```
1  // Erstellt: 23.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 8
4  // Beschreibung: Gibt e-Funktion, Sinus, Kosinus oder geometrische ↩
       Reihe in Reihendarstellung aus.
5
6  #include <iostream>
7  using namespace std;
8
9  int main() {
10   unsigned short wahlFunktion, anzahlGlieder;
11
12   // Benutzermenü
13   cout << "Welche Funktion soll ausgegeben werden?" << endl;
14   cout << "(1) e-Funktion" << endl;
15   cout << "(2) Sinusfunktion" << endl;
16   cout << "(3) Kosinusfunktion" << endl;
17   cout << "(4) Geometrische Reihe" << endl;
18   cout << "Funktion: ";
19   cin >> wahlFunktion;
20
21   // Anzahl Glieder festlegen
22   cout << "Wie viele Reihenglieder sollen ausgegeben werden? ";
23   cin >> anzahlGlieder;
24
25   switch( wahlFunktion ) {
26     case 1: { // e-Funktion
27       cout << "Es werden die ersten " << anzahlGlieder << " ↩
             Reihenglieder der e-Funktion ausgegeben." << endl;
28       for( int i = 0; i < anzahlGlieder; i++ ) {
29         if( i == 0 ) {
30           cout << "e^x = 1";
31         } else {
32           cout << " + (1/(" << i << "!))*x^" << i;
33         }
34       }
35       break;
36     }
```

```
37      case 2: { // Sinus
38        cout << "Es werden die ersten " << anzahlGlieder << " ↩
              Reihenglieder der Sinusfunktion ausgegeben." << endl;
39        for( int i = 0; i < anzahlGlieder; i++ ) {
40          int vorzeichen = i % 2;
41          if( i == 0 ) {
42            cout << "sin(x) = x";
43          } else {
44            if( vorzeichen == 0 ) {
45              cout << " + ";
46            } else {
47              cout << " - ";
48            }
49            cout << "(1/(" << 2 * i + 1 << "!))*x^" << 2 * i + 1;
50          }
51        }
52        break;
53      }
54      case 3: { // Kosinus
55        cout << "Es werden die ersten " << anzahlGlieder << " ↩
              Reihenglieder der Kosinusfunktion ausgegeben." << endl;
56        for( int i = 0; i < anzahlGlieder; i++ ) {
57          int vorzeichen = i % 2;
58          if( i == 0 ) {
59            cout << "cos(x) = 1";
60          } else {
61            if( vorzeichen == 0 ) {
62              cout << " + ";
63            } else {
64              cout << " - ";
65            }
66            cout << "(1/(" << 2 * i << "!))*x^" << 2 * i;
67          }
68        }
69        break;
70      }
71      case 4: { // geometrische Reihe
72        cout << "Keine Funktion angegeben. Es werden die ersten " << ↩
              anzahlGlieder << " Glieder der geometrischen Reihe ↩
              ausgegeben." << endl;
73        for( int i = 0; i < anzahlGlieder; i++ ) {
74          if( i == 0 ) {
75            cout << "1/(1-x) = 1";
76          } else {
77            cout << " + x^" << i;
78          }
79        }
80        break;
81      }
82      default: { // Default: geometrische Reihe
83        cout << "Ungueltige Eingabe! Es werden die ersten " << ↩
              anzahlGlieder << " Glieder der geometrischen Reihe ↩
              ausgegeben." << endl;
84        for( int i = 0; i < anzahlGlieder; i++ ) {
85          if( i == 0 ) {
86            cout << "1/(1-x) = 1";
87          } else {
88            cout << " + x^" << i;
89          }
```

```
90        }
91        break;
92      }
93    }
94
95    cout << endl;
96
97    return 0;
98  }
```

Aufgabe 9 09.cpp

```
1  // Erstellt: 28.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 9
4  // Beschreibung: Gibt e-Funktion, Sinus, Kosinus oder geometrische ↩
       Reihe in Reihendarstellung aus. In dieser Version erfolgt die ↩
       Ausgabe über vordefinierte Funktionen.
5
6  #include <iostream>
7  using namespace std;
8
9  // Funktionsprototypen
10 void sinus( const int anzahlGlieder );
11 void kosinus( const int anzahlGlieder );
12 void eFunktion( const int anzahlGlieder );
13 void geomReihe( const int anzahlGlieder );
14
15 // Hauptfunktion
16 int main() {
17   unsigned short wahlFunktion, anzahlGlieder;
18
19   // Benutzermenü
20   cout << "Welche Funktion soll ausgegeben werden?" << endl;
21   cout << "(1) e-Funktion" << endl;
22   cout << "(2) Sinusfunktion" << endl;
23   cout << "(3) Kosinusfunktion" << endl;
24   cout << "(4) Geometrische Reihe" << endl;
25   cout << "Funktion: ";
26   cin >> wahlFunktion;
27
28   cout << "Wie viele Reihenglieder sollen ausgegeben werden? ";
29   cin >> anzahlGlieder;
30
31   // Hier werden jetzt nur die Funktionen aufgerufen
32   switch( wahlFunktion ) {
33     case 1: { // e-Funktion
34       eFunktion ( anzahlGlieder );
35       break;
36     }
37     case 2: { // Sinus
38       sinus( anzahlGlieder );
39       break;
40     }
41     case 3: { // Kosinus
42       kosinus( anzahlGlieder );
43       break;
44     }
45     case 4: { // geometrische Reihe
```

```
46        geomReihe( anzahlGlieder );
47        break;
48      }
49      default: { // geometrische Reihe
50        cout << "Ungueltige Eingabe! ";
51        geomReihe( anzahlGlieder );
52        break;
53      }
54    }
55
56    cout << endl;
57
58    return 0;
59  }
60
61  // Funktionsimplementierungen
62
63  void eFunktion( const int anzahlGlieder ) {
64    cout << "Es werden die ersten " << anzahlGlieder << " Reihenglieder ↩
        der e-Funktion ausgegeben." << endl;
65    for( int i = 0; i < anzahlGlieder; i++ ) {
66      if ( i == 0 ) {
67        cout << "e^x = 1";
68      } else {
69        cout << " + (1/(" << i << "!))*x^" << i;
70      }
71    }
72  }
73
74  void sinus( const int anzahlGlieder ) {
75    cout << "Es werden die ersten " << anzahlGlieder << " Reihenglieder ↩
        der Sinusfunktion ausgegeben." << endl;
76    for( int i = 0; i < anzahlGlieder; i++ ) {
77      int vorzeichen = i % 2;
78      if( i == 0 ) {
79        cout << "sin(x) = x";
80      } else {
81        if( vorzeichen == 0 ) {
82          cout << " + ";
83        } else {
84          cout << " - ";
85        }
86        cout << "(1/(" << 2 * i + 1 << "!))*x^" << 2 * i + 1;
87      }
88    }
89  }
90
91  void kosinus( const int anzahlGlieder ) {
92    cout << "Es werden die ersten " << anzahlGlieder << " Reihenglieder ↩
        der Kosinusfunktion ausgegeben." << endl;
93    for( int i = 0; i < anzahlGlieder; i++ ) {
94      int vorzeichen = i % 2;
95      if( i == 0 ) {
96        cout << "cos(x) = 1";
97      } else {
98        if( vorzeichen == 0 ) {
99          cout << " + ";
100       } else {
101         cout << " - ";
```

```
102       }
103       cout << "(1/(" << 2 * i << "!))*x^" << 2 * i;
104     }
105   }
106 }
107
108 void geomReihe( const int anzahlGlieder ) {
109   cout << "Es werden die ersten " << anzahlGlieder << " Glieder der ↩
        geometrischen Reihe ausgegeben." << endl;
110   for( int i = 0; i < anzahlGlieder; i++ ) {
111     if( i == 0 ) {
112       cout << "1/(1-x) = 1";
113     } else {
114       cout << " + x^" << i;
115     }
116   }
117 }
```

Aufgabe 10 10.cpp

```
1  // Erstellt: 28.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 10
4  // Beschreibung: Berechnet die Fakultaet einer Zahl.
5
6
7  #include <iostream>
8  using namespace std;
9
10 // Prototyp der Fakultaetsfunktion
11 int faculty( const int zahl );
12
13 //Dies ist die Hauptfunktion
14 int main() {
15    int zahl=0;
16
17    cout << "Bitte eine Ganzzahl eingeben: ";
18    cin >> zahl;
19
20    if( zahl >= 0 ) {
21       cout << zahl << "! = " << faculty( zahl ) << endl;
22    } else {
23       cout << "FEHLER: Keine negativen Zahlen eingeben!" << endl;
24    }
25
26    return 0;
27 }
28
29 // Fakultaet berechnen und Ergebnis zurueckgeben - nicht-rekursive ↩
      Version
30 int faculty( const int zahl ) {
31   int ergebnis = zahl;
32
33   if( zahl == 0 ) {
34     ergebnis = 1;
35   }
36
37   while( zahl > 1 ) {
38     ergebnis *= --zahl;
```

```
39    }
40
41    return ergebnis;
42  }
43
44  // Alternative Fakultaetsfunktion (rekursiv)
45  // int faculty( const int zahl ) {
46  //     if( zahl > 1 ) {
47  //         return ( faculty( zahl - 1 ) * zahl );
48  //     } else {
49  //         return 1;
50  //     }
51  // }
```

Aufgabe 11 11.cpp

```
1  // Erstellt: 28.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 11
4  // Beschreibung: Quadriert eine eingegebene Zahl. Der Funktionsaufruf ↩
       erfolgt mit einem Zeiger.
5
6
7  #include <iostream>
8  using namespace std;
9
10 // Funktionsprototyp
11 void quadrieren( int* zahl );
12
13 //Dies ist die Hauptfunktion
14 int main() {
15   int zahl;
16
17   cout << "Bitte geben Sie eine Zahl ein: ";
18   cin >> zahl;
19   cout << zahl << "^2 ergibt ";
20
21   quadrieren( &zahl ); // Call by Pointer
22
23   cout << zahl << endl;
24
25   return 0;
26 }
27
28 // Quadrieren eines Zeigers
29 void quadrieren( int* zahl ) {
30   *zahl *= *zahl; // *zahl ist Dereferenzierung, also Zurückgreifen ↩
         auf den Wert
31 }
```

Aufgabe 12 12.cpp

```
1  // Erstellt: 28.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 12
4  // Beschreibung: Quadriert eine eingegebene Zahl. Der Funktionsaufruf ↩
       erfolgt mit einer Referenz.
5
```

```
6
7  #include <iostream>
8  using namespace std;
9
10 // Funktionsprototyp
11 void quadrieren( int& zahl );
12
13 //Dies ist die Hauptfunktion
14 int main() {
15   int zahl;
16
17   cout << "Bitte geben Sie eine Zahl ein: ";
18   cin >> zahl;
19   cout << zahl << "^2 ergibt ";
20
21   quadrieren( zahl ); // Call by Reference
22
23   cout << zahl << endl;
24
25   return 0;
26 }
27
28
29 // Quadrieren einer Referenz
30 void quadrieren( int& zahl ) {
31   zahl *= zahl;
32 }
```

Aufgabe 13 13.cpp

```
1  // Erstellt: 31.03.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 13
4  // Beschreibung: Multipliziert zwei komplexe Zahlen miteinander.
5
6
7  #include <iostream>
8  using namespace std;
9
10 // Funktionsprototyp
11 void multiplizieren( int* zahl1, int* zahl2, int* ergebnis );
12
13 //Dies ist die Hauptfunktion
14 int main() {
15   int zahl1[2];
16   int zahl2[2];
17   int ergebnis[2];
18
19   cout << "Bitte geben Sie den Realteil der ersten Zahl ein: ";
20   cin >> zahl1[0];
21   cout << "Bitte geben Sie den Imaginaerteil der ersten Zahl ein: ";
22   cin >> zahl1[1];
23
24   cout << "Bitte geben Sie den Realteil der ersten Zahl ein: ";
25   cin >> zahl2[0];
26   cout << "Bitte geben Sie den Imaginaerteil der ersten Zahl ein: ";
27   cin >> zahl2[1];
28
29   multiplizieren( zahl1, zahl2, ergebnis );
```

```
30
31   cout << "zahl1 * zahl2 = " << ergebnis[0] << " + j" << ergebnis[1] ↩
        << endl;
32
33   return 0;
34 }
35
36
37 void multiplizieren( int* zahl1, int* zahl2, int* ergebnis ) {
38   ergebnis[0] = zahl1[0] * zahl2[0] - zahl1[1] * zahl2[1];
39   ergebnis[1] = zahl1[0] * zahl2[1] + zahl1[1] * zahl2[0];
40 }
```

Aufgabe 14 14.cpp

```
1  // Erstellt: 31.03.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 14
4  // Beschreibung: Multipliziert zwei 3x3 Matrizen miteinander.
5
6
7  #include <iostream>
8  using namespace std;
9
10 const int ZEILEN = 3;
11 const int SPALTEN = 3;
12
13 // Funktionsprototypen
14 void matrixEingeben( int eingabeMatrix[ZEILEN][SPALTEN] );
15 void matrixAusgeben( int matrix[ZEILEN][SPALTEN] );
16 void multiplizieren( int matrixA[ZEILEN][SPALTEN],
17                      int matrixB[ZEILEN][SPALTEN],
18                      int matrixErgebnis[ZEILEN][SPALTEN] );
19
20 //Dies ist die Hauptfunktion
21 int main() {
22   int matrixA[ZEILEN][SPALTEN];
23   int matrixB[ZEILEN][SPALTEN];
24   int matrixErgebnis[ZEILEN][SPALTEN];
25
26   // Matrizen eingeben
27   cout << "Matrix A eingeben:" << endl;
28   matrixEingeben( matrixA );
29   cout << "Matrix B eingeben:" << endl;
30   matrixEingeben( matrixB );
31
32   // Produkt berechnen
33   multiplizieren( matrixA, matrixB, matrixErgebnis );
34
35   // Ergebnis ausgeben
36   cout << "A * B:" << endl;
37   matrixAusgeben( matrixErgebnis );
38
39   return 0;
40 }
41
42 // Funktionsimplementierungen
43
```

```
44 // Funktion, mit der eine Matrix von der Kommandozeile eingelesen werden kann
45 void matrixEingeben( int eingabeMatrix[ZEILEN][SPALTEN] ) {
46   // Matrix elementeweise füllen
47   for( int i = 0; i < ZEILEN; i++ ) {
48     for( int j = 0; j < SPALTEN; j++ ) {
49       cout << "Element a" << i+1 << j+1 << " eingeben: ";
50       cin >> eingabeMatrix[i][j];
51     }
52   }
53
54   // Eingegebene Matrix zur Veranschaulichung ausgeben
55   cout << "Eingegebene Matrix:" << endl;
56   matrixAusgeben( eingabeMatrix );
57   cout << endl;
58 }
59
60 // Matrix elementeweise ausgeben
61 void matrixAusgeben( int matrix[ZEILEN][SPALTEN]) {
62   for( int i = 0; i < ZEILEN; i++ ) {
63     for( int j = 0; j < SPALTEN; j++ ) {
64       cout << matrix[i][j] << " ";
65     }
66     cout << endl;
67   }
68 }
69
70 // Die Matrizenmultiplikation nach dem Falkschen Schema
71 void multiplizieren( int matrixA[ZEILEN][SPALTEN],
72                      int matrixB[ZEILEN][SPALTEN],
73                      int matrixErgebnis[ZEILEN][SPALTEN] ) {
74   for( int i = 0; i < ZEILEN; i++ ) {
75     for( int j = 0; j < SPALTEN; j++ ) {
76       matrixErgebnis[i][j] = 0;
77       for( int k = 0; k < SPALTEN; k++) {
78         matrixErgebnis[i][j] += matrixA[i][k] * matrixB[k][j];
79       }
80     }
81   }
82 }
```

Aufgabe 15 15.cpp

```
1  // Erstellt: 28.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 15
4  // Beschreibung: Eine Autoklasse mit den Eigenschaften Baujahr, Marke, Geschwindigkeit und Beschleunigung. Ein Auto kann hupen und beschleunigen.
5
6  #include <iostream>
7  #include <string>
8  using namespace std;
9
10 // Eigener Variablentyp für die Automarke
11 enum automarke { Mercedes, Audi, Volkswagen, BMW };
12
13
14 // Klasse Auto
```

```
15 class Auto {
16 private:
17   automarke marke;
18   unsigned int baujahr;
19   float beschleunigung;
20   unsigned int maxGeschwindigkeit;
21
22 public:
23   Auto(); // Standardkonstruktor
24   ~Auto(); // Standarddestruktor
25
26   // Get-Methoden - inline
27   automarke getMarke() { return marke; }
28   unsigned int getBaujahr() { return baujahr; }
29   float getBeschleunigung() { return beschleunigung; }
30   unsigned int getMaxGeschwindigkeit() { return maxGeschwindigkeit; }
31
32   // Set-Methoden - inline
33   void setMarke( automarke _marke ) { marke = _marke; }
34   void setBaujahr( int _baujahr ) { baujahr = _baujahr; }
35   void setBeschleunigung( float _beschleunigung ) { beschleunigung = ↩
         _beschleunigung; }
36   void setMaxGeschwindigkeit( int _maxGeschwindigkeit ) { ↩
         maxGeschwindigkeit = _maxGeschwindigkeit; }
37
38   void beschleunigen();
39   void hupen();
40 };
41
42 //Dies ist die Hauptfunktion
43 int main() {
44   Auto* golf = new Auto();
45
46   golf->setMarke( Volkswagen );
47   golf->setBaujahr( 2005 );
48   golf->setBeschleunigung( 7.3 );
49   golf->setMaxGeschwindigkeit( 183 );
50
51   golf->beschleunigen();
52   golf->hupen();
53
54   delete golf;
55
56   return 0;
57 }
58
59
60 // Funktionsdefinitionen der Klasse Auto
61
62 Auto::Auto() {
63   cout << "Ein neues Auto wird erstellt!" << endl;
64 }
65
66 Auto::~Auto() {
67   cout << "Ein Auto wird zerstoert!" << endl;
68 }
69
70 void Auto::beschleunigen() {
71   cout << "Beschleunige bis zur Maximalgeschwindigkeit...";
```

```
72    for( int i = 0; i < maxGeschwindigkeit; i += 10 ) {
73      cout << " ...";
74    }
75    cout << endl << maxGeschwindigkeit << " km/h erreicht." << endl;
76  }
77
78  void Auto::hupen() {
79    cout << "Tuutuuuuut!" << endl;
80  }
```

Aufgabe 16 16.cpp

```
1  // Erstellt: 28.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 16
4  // Beschreibung: Verwendung der String-Klasse
5
6  // Ich studiere gerne an der Universitaet Karlsruhe. -> wird ersetzt ↩
       durch:
7  // Ich studiere Elektro- und Informaionstechnik am Karlsruher Institut↩
       fuer Technologie.
8
9  #include <iostream>
10 #include <string>
11 using namespace std;
12
13
14 //Dies ist die Hauptfunktion
15 int main() {
16   string satz = "Ich studiere gerne an der Universitaet Karlsruhe.";
17   cout << satz << endl;
18
19   cout << "Etwas Manipulation..." << endl;
20
21   /* Ersetze "gerne an der Universitaet Karlsruhe."
22      durch "Elektro- und Informationstechnik am Karlsruher
23      Institut fuer Technologie. :)" */
24   size_t position1, position2;
25   position1 = satz.find( "gerne" );
26   position2 = satz.find( "Karlsruhe" ) + 10; // Anfangsbuchstabe 'K' ↩
       plus Länge plus Punkt
27   satz.replace( position1, position2 - position1, "Elektro- und ↩
       Informationstechnik am Karlsruher Institut fuer Technologie. :)"↩
        );
28   cout << satz << endl;
29
30   cout << "Nochmal etwas Manipulation..." << endl;
31
32   // "seit 2 Semestern " vor "Elektro" einfügen
33   position1 = satz.find( "Elektro" );
34   satz.insert( position1, "seit 2 Semestern " );
35   cout << satz << endl;
36
37   cout << "Das Smiley wirkt etwas unprofessionell..." << endl;
38
39   // Smiley suchen und entfernen
40   position1 = satz.rfind( ":)" ) - 1;
41   satz.erase( position1, 3 );
42   cout << satz << endl;
```

```
43
44    return 0;
45 }
```

Aufgabe 17 main.cpp

```
1  // Erstellt: 28.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 17
4  // Beschreibung: Ableitung von Autos, Motorrädern und LKWs von ↩
       Fahrzeugen
5
6  //main.cpp:
7  #include "fahrzeug.h"
8  #include "auto.h"
9  #include "lkw.h"
10 #include "motorrad.h"
11
12 #include <iostream>
13 using namespace std;
14
15 int main() {
16   Fahrzeug* meinFahrzeug = 0; // allgemeines Fahrzeug
17
18   // Benutzermenü
19   unsigned short auswahl;
20   cout << "Was fuer ein Fahrzeug soll erstellt werden?" << endl;
21   cout << "(1) Auto" << endl;
22   cout << "(2) LKW" << endl;
23   cout << "(3) Motorrad" << endl;
24   cout << "Eingabe: ";
25   cin >> auswahl;
26
27   // Den richtigen Fahrzeugtyp erzeugen, dynamisch erzeugtes Objekt
28   switch( auswahl ) {
29     case 1: {
30       meinFahrzeug = new Auto;
31       break;
32     }
33     case 2: {
34       meinFahrzeug = new LKW;
35       break;
36     }
37     case 3: {
38       meinFahrzeug = new Motorrad;
39       break;
40     }
41     default: {
42       meinFahrzeug = new Fahrzeug;
43     }
44   }
45
46   // Puffervariablen
47   unsigned short baujahr, maxGeschwindigkeit;
48   automarke marke;
49   float beschleunigung;
50
51   // Eingabe der Automarke
52   cout << "Marke eingeben:" << endl;
```

```
53    cout << "(1) Mercedes" << endl;
54    cout << "(2) Audi" << endl;
55    cout << "(3) Volkswagen" << endl;
56    cout << "(4) BMW" << endl;
57    cout << "(5) Andere" << endl;
58    cout << "Eingabe: ";
59    cin >> auswahl;
60
61    // Zuweisung der korrekten Automarke
62    switch( auswahl ) {
63      case 1: {
64        marke = Mercedes;
65        break;
66      }
67      case 2: {
68        marke = Audi;
69        break;
70      }
71      case 3: {
72        marke = Volkswagen;
73        break;
74      }
75      case 4: {
76        marke = BMW;
77        break;
78      }
79      default: {
80        marke = unbekannt;
81      }
82    }
83
84    // Weitere Fahrzeugparameter eingeben
85    cout << "Baujahr eingeben: ";
86    cin >> baujahr;
87    cout << "Hoechstgeschwindigkeit eingeben: ";
88    cin >> maxGeschwindigkeit;
89    cout << "Beschleunigung von 0 auf 100 km/h in Sekunden eingeben: ";
90    cin >> beschleunigung;
91
92    // Puffervariablen an Fahrzeug zuweisen
93    meinFahrzeug->setBaujahr( baujahr );
94    meinFahrzeug->setMarke( marke );
95    meinFahrzeug->setBeschleunigung( beschleunigung );
96    meinFahrzeug->setMaxGeschwindigkeit( maxGeschwindigkeit );
97
98    // Steckbrief ausgeben, beschleunigen, hupen
99    cout << endl << "Steckbrief:" << endl;
100   meinFahrzeug->steckbrief();
101   cout << endl;
102   meinFahrzeug->beschleunigen();
103   meinFahrzeug->hupen();
104   cout << endl;
105
106   delete meinFahrzeug; // reservierten Speicher wieder frei geben
107   return 0;
108 }
```

fahrzeug.h

```cpp
#ifndef FAHRZEUG_H
#define FAHRZEUG_H

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

// Basisklasse muss Automarken kennen
enum automarke { Mercedes, Audi, Volkswagen, BMW, unbekannt };

// Basisklasse Fahrzeug
class Fahrzeug {
protected: // abgeleitete Klassen können hierauf zugreifen
  automarke marke;
  unsigned int baujahr;
  float beschleunigung;
  unsigned int maxGeschwindigkeit;

  // Einige Einträge des Steckbriefs sind in allen abgeleiteten Klassen gleich
  void allgemeinerSteckbrief();

public:
  Fahrzeug(); // Standardkonstruktor
  virtual ~Fahrzeug(); // Standarddestruktor

  // Get-Methoden - inline
  automarke getMarke() { return marke; }
  unsigned int getBaujahr() {return baujahr; }
  float getBeschleunigung() { return beschleunigung; }
  unsigned int getMaxGeschwindigkeit() { return maxGeschwindigkeit; }

  // Set-Methoden - inline
  void setMarke( automarke _marke ) { marke = _marke; }
  void setBaujahr( int _baujahr ) { baujahr = _baujahr; }
  void setBeschleunigung( float _beschleunigung ) { beschleunigung = _beschleunigung; }
  void setMaxGeschwindigkeit( int _maxGeschwindigkeit ) { maxGeschwindigkeit = _maxGeschwindigkeit; }

  void beschleunigen();
  virtual void hupen(); // Soll in abgeleiteten Klassen redefiniert werden
  virtual void steckbrief(); // Soll in abgeleiteten Klassen redefiniert werden

};


#endif // FAHRZEUG_H
```

fahrzeug.cpp

```cpp
#include "fahrzeug.h"

Fahrzeug::Fahrzeug() {
  cout << "Ein neues Fahrzeug wird erstellt!" << endl;
}
```

```
6
7  Fahrzeug::~Fahrzeug() {
8    cout << "Ein Fahrzeug wird zerstoert!" << endl;
9  }
10
11 void Fahrzeug::beschleunigen() {
12   cout << "Beschleunige bis zur Maximalgeschwindigkeit...";
13   for( unsigned int i = 0; i < maxGeschwindigkeit; i += 10 ) {
14     cout << " ...";
15   }
16   cout << endl << maxGeschwindigkeit << " km/h erreicht." << endl;
17 }
18
19 void Fahrzeug::hupen() {
20   cout << "Ich hupe!" << endl;
21 }
22
23 // Einige Einträge des Steckbriefs sind in allen abgeleiteten Klassen ↩
     gleich
24 void Fahrzeug::allgemeinerSteckbrief() {
25   string markeAusgabe;
26
27   // automarke in auszugebenden String umwandeln
28   switch( marke ) {
29     case 0: {
30       markeAusgabe = "Mercedes";
31       break;
32     }
33     case 1: {
34       markeAusgabe = "Audi";
35       break;
36     }
37     case 2: {
38       markeAusgabe = "Volkswagen";
39       break;
40     }
41     case 3: {
42       markeAusgabe = "BMW";
43       break;
44     }
45     default:
46       markeAusgabe = "unbekannt";
47   }
48
49   // Ausgabe
50   cout << markeAusgabe << endl;
51   cout << "Mein Baujahr ist " << baujahr << endl;
52   cout << "Ich komme in " << beschleunigung << " Sekunden von 0 auf ↩
       100 km/h" << endl;
53   cout << "Meine Hoechstgeschwindigkeit ist " << maxGeschwindigkeit <<↩
        " km/h" << endl;
54 }
55
56 void Fahrzeug::steckbrief() {
57   cout << "Ich bin ein Fahrzeug der Marke ";
58   allgemeinerSteckbrief();
59 }
```

auto.h

```
#ifndef AUTO_H
#define AUTO_H

#include "fahrzeug.h"

// Auto erbt von Fahrzeug
class Auto : public Fahrzeug {
public:
  Auto();
  ~Auto();

  void hupen();
  void steckbrief();
};

#endif // AUTO_H
```

auto.cpp

```
#include "auto.h"


Auto::Auto() {
  cout << "Ein neues Auto wird erstellt!" << endl;
}

Auto::~Auto() {
  cout << "Ein Auto wird zerstoert!" << endl;
}

void Auto::hupen() {
  cout << "Tuutuut!" << endl;
}

void Auto::steckbrief() {
  cout << "Ich bin ein Auto der Marke ";
  allgemeinerSteckbrief();
}
```

lkw.h

```
#ifndef LKW_H
#define LKW_H

#include "fahrzeug.h"

// LKW erbt von Fahrzeug
class LKW : public Fahrzeug {
public:
  LKW();
  ~LKW();

  void hupen();
  void steckbrief();
};

```

```
16 #endif // LKW_H
```

lkw.cpp

```
1  #include "lkw.h"
2
3
4  LKW::LKW() {
5    cout << "Ein neuer LKW wird erstellt!" << endl;
6  }
7
8  LKW::~LKW() {
9    cout << "Ein LKW wird zerstoert!" << endl;
10 }
11
12 void LKW::hupen() {
13   cout << "Troeoeoeoet!" << endl;
14 }
15
16 void LKW::steckbrief() {
17   cout << "Ich bin ein LKW der Marke ";
18   allgemeinerSteckbrief();
19 }
```

motorrad.h

```
1  #ifndef MOTORRAD_H
2  #define MOTORRAD_H
3
4  #include "fahrzeug.h"
5
6  // Motorrad erbt von Fahrzeug
7  class Motorrad : public Fahrzeug {
8  public:
9    Motorrad();
10   ~Motorrad();
11
12   void hupen();
13   void steckbrief();
14 };
15
16 #endif // MOTORRAD_H
```

motorrad.cpp

```
1  #include "motorrad.h"
2
3
4  Motorrad::Motorrad() {
5    cout << "Ein neues Motorrad wird erstellt!" << endl;
6  }
7
8  Motorrad::~Motorrad() {
9    cout << "Ein Motorrad wird zerstoert!" << endl;
10 }
11
12 void Motorrad::hupen() {
13   cout << "Tuetuet!" << endl;
```

```
14 }
15
16 void Motorrad::steckbrief() {
17   cout << "Ich bin ein Motorrad der Marke ";
18   allgemeinerSteckbrief();
19 }
```

Aufgabe 18 main.cpp

```
1  // Erstellt: 28.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 18
4  // Beschreibung: Auslesen von Fahrzeugobjekten aus einer Textdatei, ↩
       welche dann einem Vektor 'garage' gespeichert werden.
5
6  #include "fahrzeug.h"
7  #include "auto.h"
8  #include "lkw.h"
9  #include "motorrad.h"
10 #include "garage.h"
11
12 #include <iostream>
13 using namespace std;
14
15 int main() {
16   Garage* meineGarage = new Garage; // neue Garage anlegen
17
18   meineGarage->setDateiname( "garage.txt" ); // Dateiname setzen
19   meineGarage->dateiEinlesen(); // Datei einlesen lassen
20   meineGarage->angeben(); // Garageninhalt ausgeben
21
22   delete meineGarage; // Reservierten Speicher freigeben
23
24   return 0;
25 }
```

garage.h

```
1  #ifndef GARAGE_H_INCLUDED
2  #define GARAGE_H_INCLUDED
3
4  #include "fahrzeug.h"
5  #include "auto.h"
6  #include "motorrad.h"
7  #include "lkw.h"
8
9  #include <string>
10 #include <vector>
11 using namespace std;
12
13 class Garage{
14 private:
15   vector<Fahrzeug*> fahrzeuge;
16   string dateiname;
17
18 public:
19   Garage();
20   ~Garage();
```

```
21
22    string getFateiname() { return dateiname; }
23    void setDateiname( string _dateiname ) { dateiname = _dateiname; }
24
25    void dateiEinlesen(); // Fuellt die Garage mit den Fahrzeugen aus ↩
          der Textdatei
26    void angeben(); // Gibt den Inhalt der Garage aus
27 };
28
29 #endif // GARAGE_H_INCLUDED
```

garage.cpp

```
1  #include "garage.h"
2
3  #include <fstream>
4  #include <iostream>
5  #include <string>
6  #include <cstdlib>
7  using namespace std;
8
9  Garage::Garage() {
10 }
11
12 Garage::~Garage() {
13 }
14
15 void Garage::dateiEinlesen() {
16   ifstream datei( dateiname.c_str() );
17   if( datei ){ // Einlesen erfolgreich?
18     string zeile; // Puffervariable
19     while( !datei.eof() ) {
20       getline( datei, zeile );
21       size_t raute = zeile.find( "#" );
22       size_t eckeAuf = zeile.find( "[" );
23       size_t doppelpunkt = zeile.find( ":" );
24
25       if( raute == string::npos ) { // kein Kommentar?
26         if( eckeAuf != string::npos ) { // '[' leitet neues Fahrzeug ↩
              ein
27           size_t eckeZu = zeile.find( "]" );
28           string fahrzeugtyp = zeile.substr( eckeAuf+1, eckeZu - ↩
                eckeAuf - 1 );
29           // Fahrzeugtyp feststellen
30           if( fahrzeugtyp == "LKW" ) {
31             fahrzeuge.push_back( new LKW );
32           } else if( fahrzeugtyp == "Auto" ) {
33             fahrzeuge.push_back( new Auto );
34           } else if( fahrzeugtyp == "Motorrad" ) {
35             fahrzeuge.push_back( new Motorrad );
36           } else {
37             fahrzeuge.push_back( new Fahrzeug );
38           }
39         } else if( zeile.find( "Marke:" ) != string::npos ) { // Zeile↩
              für Marke
40           string markenstring = zeile.substr( 7, zeile.length() - 7 );
41           if( markenstring == "Mercedes" ) {
42             fahrzeuge.back()->setMarke( Mercedes );
43           } else if( markenstring == "Audi" ){
```

```
44              fahrzeuge.back()->setMarke( Audi );
45            } else if( markenstring == "Volkswagen" ){
46              fahrzeuge.back()->setMarke( Volkswagen );
47            } else if( markenstring == "BMW" ){
48              fahrzeuge.back()->setMarke( BMW );
49            } else {
50              fahrzeuge.back()->setMarke( unbekannt );
51            }
52          } else if( zeile.find( "Baujahr:" ) != string::npos ) { // ↩
                Zeile für Baujahr
53            string baujahrstring = zeile.substr( 9, zeile.length() - 7 )↩
                ;
54            fahrzeuge.back()->setBaujahr( atoi( baujahrstring.c_str() ) ↩
                );
55          } else if( zeile.find( "Geschwindigkeit:" ) != string::npos ) ↩
                { // Zeile für Geschwindigkeit
56            string geschwindigkeitstring = zeile.substr( 17, zeile.↩
                length() - 17 );
57            fahrzeuge.back()->setMaxGeschwindigkeit( atoi( ↩
                geschwindigkeitstring.c_str() ) );
58          } else if( zeile.find( "Beschleunigung:" ) != string::npos ) {↩
                // Zeile für Beschleunigung
59            string beschleunigungstring = zeile.substr( 15, zeile.length↩
                () - 15 );
60            fahrzeuge.back()->setBeschleunigung( atof( ↩
                beschleunigungstring.c_str() ) );
61          }
62        }
63      }
64    }
65  }
66
67  void Garage::angeben() {
68    cout << "Die Garage stellt sich vor:" << endl << endl;
69    for( int i = 0; i < fahrzeuge.size(); i++) {
70      fahrzeuge[i]->steckbrief();
71      fahrzeuge[i]->hupen();
72      cout << endl;
73    }
74  }
```