Karlsruher Institut für Technologie
Institut für Technik der Informationsverarbeitung

Aufgaben zum C++ Tutorium

Informationstechnik
Tutorium und Praktikum

Andreas Lauber, Marc Weber, Jijing Yan

Institutsleitung
Prof. Dr.-Ing. Dr. h.c. J. Becker
Prof. Dr.-Ing. E. Sax
Prof. Dr. rer. nat. W. Stork

KIT - Die Forschungsuniversität 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.

Um Ihnen ein Gefühl zu geben, ob Sie zu schnell oder zu langsam sind, haben wir die folgenden Termine für die Bearbeitung der Aufgaben vorgesehen:

- Kapitel 1: Einstieg in die Rechnerarchitektur (Tutorium 1)
- Kapitel 2: Einstieg in C++ (Tutorium 1 - 4)
  - Tutorium 1: Aufgabe 1 - 3
  - Tutorium 2: Aufgabe 4 - 8
  - Tutorium 3: Aufgabe 8 - 11
  - Tutorium 4: Aufgabe 12 - 13
- Kapitel 3: Einstieg in die hardwarenahe Programmierung (Tutorium 4 - 5)

Die Aufgabenverteilung können Sie auch frei und unabhängig von den angegebenen Tutoriumsterminen bearbeiten. Es soll Ihnen lediglich als Anhaltspunkt dienen. Sollten Sie die Aufgaben in der angegebenen Zeit nicht lösen können, bitten wir Sie die Tutorien entsprechend Vor- und Nachzubereiten.

Verbesserungsvorschläge, sowie jede Art von Kritik zu diesem Dokument, sind sehr erwünscht. Bitte senden Sie diese an *marc.weber3@kit.edu*.

Kapitel 1

Einstieg in die Rechnerarchitektur

Aufgabe 1 Rechnerarchitektur

Was versteht man unter dem Begriff Rechnerarchitektur? Recherchieren Sie den Begriff.

Aufgabe 2 Von-Neumann und Harvard Architektur

Zeichnen Sie die von-Neumann- und die Harvard-Architektur und erläutern Sie der Hauptunterschied.

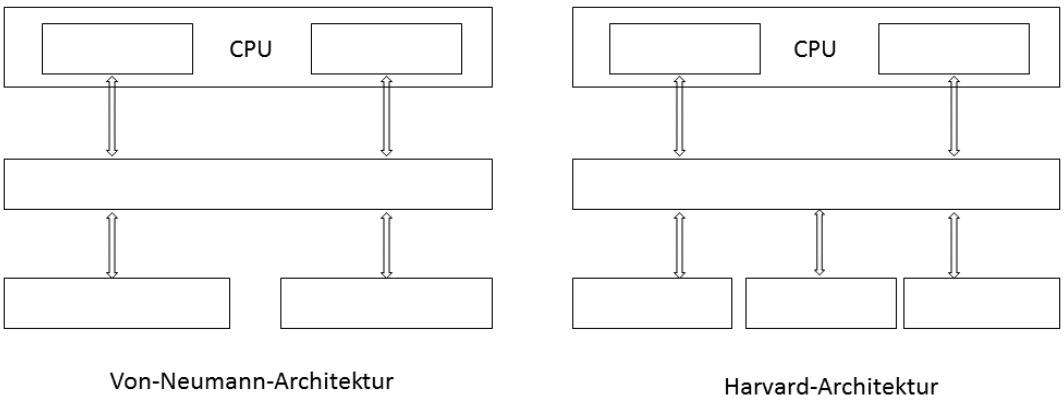


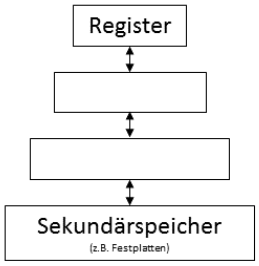
Abbildung 1.1: Rechner Architektur

Aufgabe 3 Von-Neumann-Zyklus

Was bezeichnet der Von-Neumann-Zyklus? Beschreiben Sie die fünf ablaufenden Teilschritten, schauen Sie sich hierfür das Video unter: `https://www.youtube.com/watch?v=LBagfbOy_gs` an.

Aufgabe 4 Speicherhirarchie

Recherchieren Sie den Begriff Speicherhierarchie und vervollständigen Sie die Abbildung 1.2. Erklären Sie noch das Lokalitätsprinzip in der Speicherhierarchie.



	Größe (ca.)	Zugriffszeit (ca.)	Technologie
Register	1 KB	250 ps	SRAM
Sekundärspeicher (z.B. Festplatten)			

Abbildung 1.2: Speicherhierarchie

Aufgabe 5 Eingebettete Systeme

Recherchieren Sie Mikrocontroller und Eingebettete Systeme. Was versteht man unter der Eingebetteten Systeme? In der Abbildung 1.3 wird beispielsweise eine Struktur für Mikrocontroller für die Entwicklung eingebetteter Systeme gezeichnet. Recherchieren Sie A/D Wandler, PWM, GPIO, Watchdog und Timer.

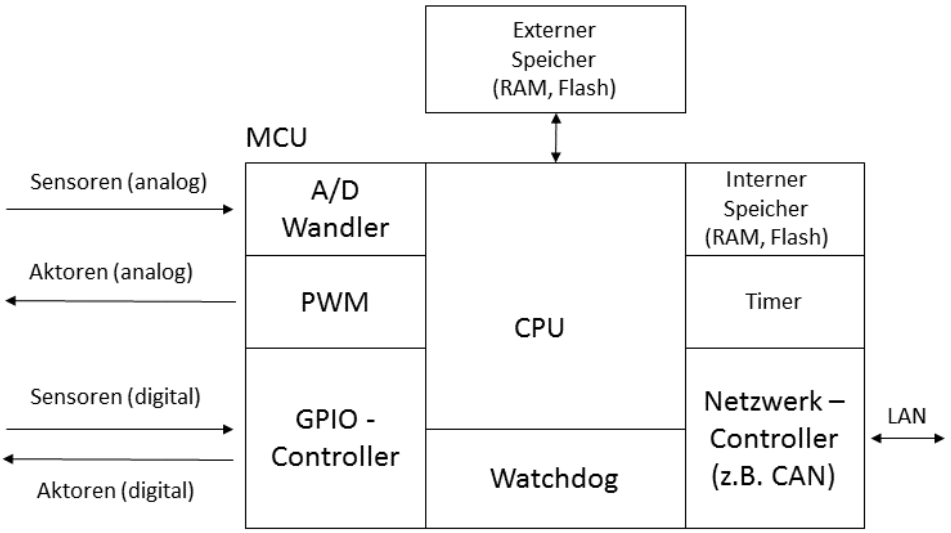


Abbildung 1.3: Struktur für Mikrocontroller

Kapitel 2

Einstieg in C++

Aufgabe 1 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 2 Wo liegt die Zahl genau?

Vorwissen:
Kompendium: Kapitel 3.4
Richtlinien: Kapitel 2

Erweitern Sie Ihr Programm aus Aufgabe 1. 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 3 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 4.
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 4 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.
3. Ändern Sie die Funktion so ab, dass negative Eingaben eine Fehlermeldung erzeugen.

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 5 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 6 Quadrieren mit Übergabe als Referenz

Vorwissen:
Kompendium: Kapitel 4.5.2
Richtlinien: Kapitel 3.5

Schreiben Sie nun ihre Funktion aus Aufgabe 5 so um, dass ihr eine Referenz als Parameter übergeben werden kann. Rufen Sie in Ihrem Hauptprogramm die neue Funktion auf.

Aufgabe 7 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 8 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 9 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 Informationstechnik 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 10 Ein paar Fahrzeuge

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

Erzeugen Sie zunächst eine Fahrzeugsklasse nach dem Beispiel der Autoklasse in Aufgabe 8. 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 11 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 8 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.

Aufgabe 12 Paketdienst

Vorwissen:
Kompendium: Kapitel 7,8
Richtlinien: Kapitel 4,6

E s soll eine Datenbank eines Paketdienstes zur Sendungsverfolgung von Paketen objektorientiert programmiert werden. Dabei sollen alle Adressen, welche ein Paket von der Aufgabe bis zur Zustellung durchläuft dokumentiert werden.
Hierzu sollen drei Klassen entstehen. Die erste Klasse speichert eine Adresse mit Namenszeile, Straße, PLZ, Ort und Land in entsprechenden privaten Attributen. Auch soll die Klasse eine Methode enthalten um die Daten vom Benutzer (von der Tastatur) abzufragen und eine Methode um alle Daten auszugeben. Die zweite Klasse enthält alle Daten zu einem Paket, wie zum Beispiel Sendungsnummer, Größe und Gewicht. Auch soll in dieser Klasse unter Verwendung der ersten Klasse die Absenderadresse und Zieladresse gespeichert werden. Die Zwischenadressen sollen ebenfalls unter Verwendung der ersten Klasse in einem entsprechenden dynamischen Container in der Klasse zum Paket gespeichert werden.
Die letzte Klasse beschreibt den Paketdienst. Die Klasse kann dabei eine beliebige Anzahl an Paketen speichern. Auch bietet die Klasse Methoden zum Hinzufügen eines Paketes und zum Suchen anhand der Sendungsnummer. Nachdem ein Paket über die Suche gefunden wurde, kann ebenfalls auf Wunsch des Users eine Station zu den Zwischenadressen hinzugefügt werden.

Aufgabe 13 Paketdienst 2

Vorwissen:
Kompendium: Kapitel 6,7,8
Richtlinien: Kapitel 4,6

D as Programm zum Paketdienst aus der vorherigen Aufgabe soll so erweitert werden, dass auch Paket-Eintragungen gelöscht werden können. Hierzu soll von der dritten Klasse Paketdienst eine neue Klasse abgeleitet werden, die eine zusätzliche Funktion zum Löschen eines Pakets nach Eingabe der Trackingnummer enthält. Dabei soll doppelter Code vermieden werden, indem entsprechender Code in eine neue (nicht öffentliche) Methode ausgelagert wird.

Kapitel 3

Lösung: Rechnerarchitektur

Aufgabe 1 Rechnerarchitektur

Amdahl, Brooks und Blaauw definieren 1967: „Rechnerarchitektur definiert sich aus den Attributen und dem Verhalten eines Computers, wie dieser von einem Maschinensprachenprogrammierer gesehen wird. Diese Definition umfasst den Befehlssatz, die Befehlsformate, die OP-Codes, die Adressierungsarten und alle Register und Speicher, die direkt durch einen Maschinensprachenprogrammierer verändert werden können. Die Implementation ist durch den aktuellen Hardwareaufbau, das logische Design und die Organisation der Datenpfade einer bestimmten Ausführung der Architektur definiert."

Stahlknecht und Hasenkamp verwenden folgende Definition: „die interne Struktur des Rechners, d. h. seinen Aufbau aus verschiedenen Komponenten, und die Organisation der Arbeitsabläufe im Rechner."

Aufgabe 2 Von-Neumann und Harvard Architektur

Von Neumann-Architektur: ein Speicher sowohl für Programm als auch für Daten
Harvard-Architektur: getrennte Speicher für Programm und Daten

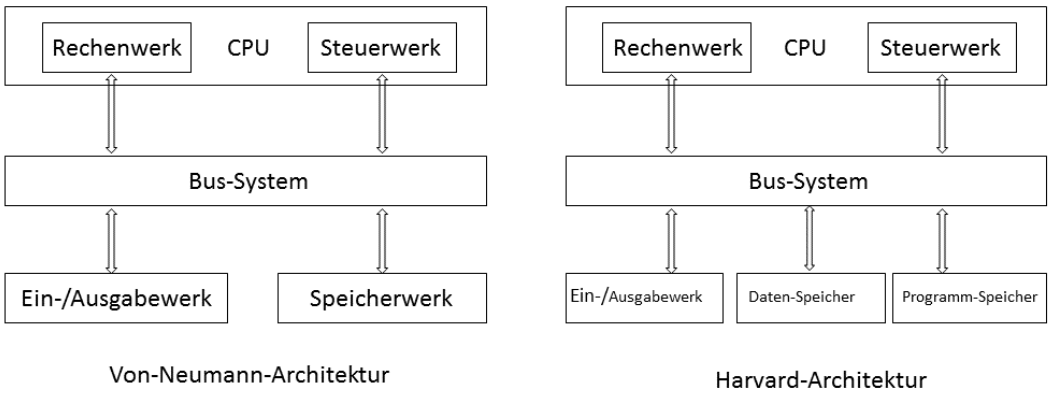


Abbildung 3.1: Rechner Architektur

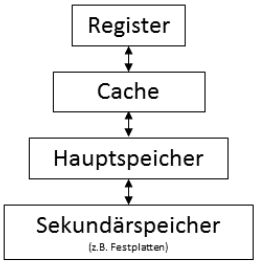
Aufgabe 3 Von-Neumann-Zyklus

Der Prozess der Befehlsverarbeitung bei Von-Neumann-Rechnern wird Von-Neumann-Zyklus genannt und besteht aus den folgenden fünf nacheinander ablaufenden Teilschritten:1. Fetch 2. Decode 3. Fetch Operands 4. Execute 5. Write Back

Aufgabe 4 Speicherhirarchie

In der Informatik bezeichnet Speicherhierarchie die Anordnung von Speichern in einer Rechnerarchitektur aus Sicht des Hauptprozessors, geordnet nach sinkender Zugriffsgeschwindigkeit, sinkenden Kosten, steigender Speicherkapazität und steigender Zugriffseinheit. (Wikipedia)

Zeitliche Lokalität: Auf Speicher, auf den kürzlich zugegriffen wurde, wird bald wieder zugegriffen (Schleifen, lokale Variablen)
Örtliche Lokalität: Auf Speicher in der Nachbarschaft von kürzlich benutzten Speicher wird bald wieder



Größe (ca.)	Zugriffszeit (ca.)	Technologie
1 KB	250 ps	SRAM
4 KB- 12 MB	1 ns	SRAM
1 GB-256 GB	100 ns	DRAM
100 GB- ...	10 ms	Magnetisch, Optisch

Abbildung 3.2: Speicherhierarchie

zugegriffen (Linearer Programmcode, Array-Zugriffe)

Aufgabe 5 Eingebettete Systeme

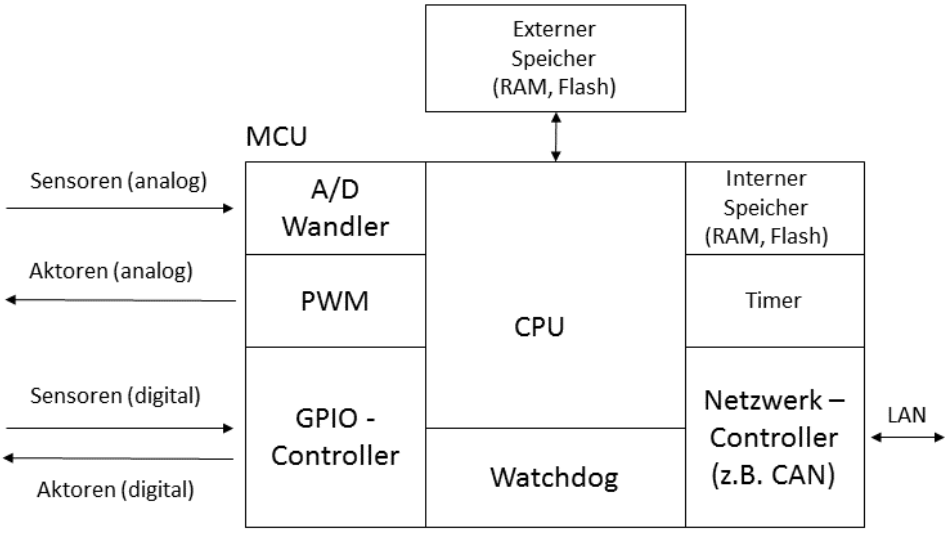


Abbildung 3.3: Struktur für Mikrocontroller

Unter eingebetteten Systemen versteht man Hard- und Softwaresysteme, die eingebettet in umgebende technische Systeme komplexe Steuerungs-, Regelungs und Datenverarbeitungsaufgaben übernehmen.

A/D- D/A Wandler: Dieser wandelt eine analoge Größe, meist eine Spannung, in einen digitalen Wert um. Das Gegenstück ist der DA-Wandler.

GPIO: General Purpose Input/Output. Sie stellen universell nutzbare Ein- und Ausgänge zur Verfügung.

Watchdog: Ein Watchdog ist eine Schaltung (extern oder im Mikrocontroller integriert), die bei einem Programmabsturz einen Reset auslöst, damit der Mikrocontroller seine Aufgabe wieder erledigen kann.

Timer: Als Timer oder auch Counter bezeichnet man bei Mikrocontrollern einen auf dem Chip integriertes Funktionsmodul, welcher beim Zählen von Ereignissen, Messen von Zeitabständen und periodischen Ausführen von Programmteilen eine wichtige Hilfe darstellt. Möchte man zum Beispiel eine Uhr realisieren, dann konfiguriert man den Timer so, dass z.B. exakt 100 mal pro Sekunde ein Interrupt ausgelöst wird. In der Interruptroutine kann man dann eine Variable hochzählen und nach 100 Interrupts die Zeitanzeige aktualisieren.

Kapitel 4

Lösungen: Einstieg in C++

Aufgabe 1 Zahlenraum.cpp

```
1  // Erstellt: 16.02.2010
2  // Autor: Felix Mauch
3  // Uebungsaufgabe 5
4  // Beschreibung: Ordnet eine eingegebene Zahl in einen Zahlenraum ein.
5
6  #include <iostream>
7  using namespace std;
8
9  int main() {
10   int zahl; // Einzugrenzende Zahl
11   char abfrage; // Abfrage nach weiterer Eingrenzung
12
13   cout << "Bitte eine ganze Zahl eingeben: ";
14   cin >> zahl;
15
16   if( zahl < 0 ) {
17     // 1. Fall: Zahl ist kleiner als 0
18     cout << "Die eingegebene Zahl ist kleiner als 0" << endl;
19   } else if( zahl == 0 ) {
20     // 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 2 ZahlenFinden.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 3 Reihen1.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 4 Reihen2.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 5 CallByPointer.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 6 CallByReference.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 7 komplexeZahlen.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 8 Autoklasse.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 9 String.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 10 Fahrzeuge

```
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

```
1  #ifndef FAHRZEUG_H
2  #define FAHRZEUG_H
3
4  #include <iostream>
5  #include <string>
6  using namespace std;
7
8  // Basisklasse muss Automarken kennen
9  enum automarke { Mercedes, Audi, Volkswagen, BMW, unbekannt };
10
11 // Basisklasse Fahrzeug
12 class Fahrzeug {
13 protected: // abgeleitete Klassen können hierauf zugreifen
14   automarke marke;
15   unsigned int baujahr;
16   float beschleunigung;
17   unsigned int maxGeschwindigkeit;
18
19   // Einige Einträge des Steckbriefs sind in allen abgeleiteten ↩
       Klassen gleich
20   void allgemeinerSteckbrief();
21
22 public:
23   Fahrzeug(); // Standardkonstruktor
24   virtual ~Fahrzeug(); // 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    virtual void hupen(); // Soll in abgeleiteten Klassen redefiniert ↩
          werden
40    virtual void steckbrief(); // Soll in abgeleiteten Klassen ↩
          redefiniert werden
41
42  };
43
44
45  #endif // FAHRZEUG_H
```

fahrzeug.cpp

```
1   #include "fahrzeug.h"
2
3   Fahrzeug::Fahrzeug() {
4     cout << "Ein neues Fahrzeug wird erstellt!" << endl;
5   }
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

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

auto.cpp

```
1  #include "auto.h"
2
3
4  Auto::Auto() {
5    cout << "Ein neues Auto wird erstellt!" << endl;
6  }
7
8  Auto::~Auto() {
9    cout << "Ein Auto wird zerstoert!" << endl;
10 }
11
12 void Auto::hupen() {
13   cout << "Tuutuut!" << endl;
14 }
```

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

lkw.h

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

```
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  }
```

Aufgabe 12 Paketdienst

```
1  //-------------------------------------------------
2  //Programmname: Paketdienst
3  //Autoren: Tobias Schwalb, Timo Sandmann, ITIV
4  //Erstellt: 09.05.2012
5  //Version: 1.1
6  //-------------------------------------------------
7
8  #include "parcelservice.h"
9
10 int main() {
11   ParcelService dhl;
12   dhl.menu();
13
14   return 0;
15 }
```

address.h

```
1  #ifndef ADDRESS_H
2  #define ADDRESS_H
3
4  #include <string>
5
6  using namespace std;
7
8  class Address {
9  protected:
10   string name;
11   string street;
12   int zip;
13   string city;
14   string country;
15 public:
16   void readIn();
17   void printAddress();
18 };
19
20 #endif // ADDRESS_H
```

address.cpp

```
1  #include <iostream>
2  #include "address.h"
3
4  void Address::readIn() {
5    cout << "Bitte Namen eingeben: ";
6    cin >> name;
7    cout << "Bitte Strasse eingeben: ";
8    cin >> street;
9    cout << "Bitte PLZ eingeben: ";
10   cin >> zip;
11   cout << "Bitte Ort eingeben: ";
```

```
12    cin >> city;
13    cout << "Bitte Land eingeben: ";
14    cin >> country;
15  }
16
17  void Address::printAddress() {
18    cout << "Name: " << name << endl;
19    cout << "Strasse: " << street << endl;
20    cout << "PLZ: " << zip << endl;
21    cout << "Ort: " << city << endl;
22    cout << "Land: " << country << endl;
23  }
```

parcel.h

```
1  #ifndef PARCEL_H
2  #define PARCEL_H
3
4  #include <vector>
5  #include "address.h"
6
7  using namespace std;
8
9  class Parcel {
10 protected:
11   int trackingnumber;
12   int size;
13   double weight;
14   Address sender;
15   Address destination;
16   vector<Address*> stations;
17 public:
18   ~Parcel();
19   void addAddress();
20   void readIn();
21   void outputDetails();
22   inline int getTrackingNumber() {
23     return trackingnumber;
24   }
25 };
26
27 #endif // PARCEL_H
```

parcel.cpp

```
1  #include <iostream>
2  #include "parcel.h"
3
4  void Parcel::addAddress() {
5    Address* newAddress = new Address();
6    newAddress->readIn();
7    stations.push_back( newAddress );
8  }
9
10 void Parcel::readIn() {
11   cout << "Bitte Trackingnummer eingeben: ";
12   cin >> trackingnumber;
13   cout << "Bitte Groessenklasse eingeben: ";
```

```
14    cin >> size;
15    cout << "Bitte Gewichtsklasse eingeben: ";
16    cin >> weight;
17    cout << endl << "Bitte Absender eingeben " << endl;
18    sender.readIn();
19    cout << endl << "Bitte Empfaenger eingeben " << endl;
20    destination.readIn();
21  }
22
23  void Parcel::outputDetails() {
24    cout << "Trackingnummer: " << trackingnumber << endl;
25    cout << "Groessenklasse: " << size << endl;
26    cout << "Gewichtsklasse: " << weight << endl;
27
28    cout << endl << "Absender : " << endl;
29    sender.printAddress();
30
31    cout << endl << "Empfaenger : " << endl;
32    destination.printAddress();
33
34    cout << endl << "Stationen : " << endl;
35    for( unsigned int i = 0; i < stations.size(); ++i ) {
36      if( stations[i] != NULL ) {
37        stations[i]->printAddress();
38        cout << endl;
39      }
40    }
41  }
42
43  Parcel::~Parcel() {
44    for( unsigned int i = 0; i < stations.size(); ++i ) {
45      delete stations[i];
46    }
47  }
```

parcelservice.h

```
1  #ifndef PARCELSERVICE_H
2  #define PARCELSERVICE_H
3
4  #include <string>
5  #include <vector>
6  #include "parcel.h"
7
8  using namespace std;
9
10 class ParcelService {
11 protected:
12   vector<Parcel*> parcels;
13 public:
14   void menu();
15   void addParcel();
16   void showAllParcels();
17   void searchParcel();
18   ~ParcelService();
19 };
20
21 #endif // PARCELSERVICE_H
```

parcelservice.cpp

```
1  #include <iostream>
2  #include "parcelservice.h"
3
4  void ParcelService::addParcel() {
5    Parcel* newParcel = new Parcel();
6    newParcel->readIn();
7
8    char antwort = ' ';
9    do {
10     cout << endl << "Moechten Sie eine (weitere) Station hinzufuegen? (J)a/(N)ein ";
11     cin >> antwort;
12     antwort = toupper(antwort);
13     if( antwort == 'J' ) {
14       newParcel->addAddress();
15     }
16   } while( antwort == 'J' );
17
18   parcels.push_back( newParcel );
19 }
20
21 void ParcelService::showAllParcels() {
22   for( unsigned int i = 0; i < parcels.size(); i++ ) {
23     if(  parcels[i] != NULL ) {
24       cout << "PAKET " << i << endl;
25       parcels[i]->outputDetails();
26       cout << endl << endl;
27     }
28   }
29 }
30
31 void ParcelService::searchParcel() {
32   int number;
33   cout << "Bitte geben Sie die Trackingnummer ein: ";
34   cin >> number;
35   unsigned int i;
36   for( i = 0; i < parcels.size(); i++ ) {
37     if( parcels[i] != NULL && parcels[i]->getTrackingNumber() == number ) {
38       cout << "PAKET " << i << endl;
39       parcels[i]->outputDetails();
40       cout << endl << endl;
41       break;
42     }
43   }
44   if( i < parcels.size() ) {
45     char antwort =  ' ';
46     do {
47       cout << endl << "Moechten Sie eine (weitere) Station hinzufuegen? (J)a/(N)ein ";
48       cin >> antwort;
49       antwort = toupper(antwort);
50       if( antwort == 'J' ) {
51         parcels[i]->addAddress();
52       }
53     } while( antwort == 'J' );
54   } else {
```

```
55      cout << "Trackingnummer nicht gefunden" << endl;
56    }
57  }
58
59  void ParcelService::menu() {
60    int auswahl = 0;
61    while( true ) {
62      cout << endl << endl;
63      cout << "MENU " << endl;
64      cout << "(1) Paket hinzufuegen" << endl;
65      cout << "(2) Alle Pakete anzeigen" << endl;
66      cout << "(3) Pakete suchen" << endl;
67      cout << "(4) Exit" << endl;
68      cout << "Bitte waehlen: ";
69
70      cin >> auswahl;
71      switch( auswahl ) {
72      case 1: addParcel(); break;
73      case 2: showAllParcels(); break;
74      case 3: searchParcel(); break;
75      default: return;
76      }
77    };
78  }
79
80  ParcelService::~ParcelService() {
81    for( unsigned int i = 0; i < parcels.size(); i++ ) {
82      delete parcels[i];
83    }
84  }
```

Aufgabe 13 Paketdienst 2

```
1  //-------------------------------------------------
2  //Programmname: Paketdienst
3  //Autoren: Tobias Schwalb, Timo Sandmann, ITIV
4  //Erstellt: 09.05.2012
5  //Version: 1.1
6  //-------------------------------------------------
7
8  #include "ParcelServiceV2.h"
9
10 int main() {
11   ParcelServiceV2 ups;
12   ups.menu();
13   return 0;
14 }
```

address.h

```
1  #ifndef ADDRESS_H
2  #define ADDRESS_H
3
4  #include <string>
5
6  using namespace std;
7
8  class Address {
```

```
9  protected:
10   string name;
11   string street;
12   int zip;
13   string city;
14   string country;
15 public:
16   void readIn();
17   void printAddress();
18 };
19
20 #endif // ADDRESS_H
```

address.cpp

```
1  #include <iostream>
2  #include "address.h"
3
4  void Address::readIn() {
5    cout << "Bitte Namen eingeben: ";
6    cin >> name;
7    cout << "Bitte Strasse eingeben: ";
8    cin >> street;
9    cout << "Bitte PLZ eingeben: ";
10   cin >> zip;
11   cout << "Bitte Ort eingeben: ";
12   cin >> city;
13   cout << "Bitte Land eingeben: ";
14   cin >> country;
15 }
16
17 void Address::printAddress() {
18   cout << "Name: " << name << endl;
19   cout << "Strasse: " << street << endl;
20   cout << "PLZ: " << zip << endl;
21   cout << "Ort: " << city << endl;
22   cout << "Land: " << country << endl;
23 }
```

parcel.h

```
1  #ifndef PARCEL_H
2  #define PARCEL_H
3
4  #include <vector>
5  #include "address.h"
6
7  using namespace std;
8
9  class Parcel {
10 protected:
11   int trackingnumber;
12   int size;
13   double weight;
14   Address sender;
15   Address destination;
16   vector<Address*> stations;
17 public:
```

```
18    ~Parcel();
19    void addAddress();
20    void readIn();
21    void outputDetails();
22    inline int getTrackingNumber() {
23      return trackingnumber;
24    }
25  };
26
27  #endif // PARCEL_H
```

parcel.cpp

```
1   #include <iostream>
2   #include "parcel.h"
3
4   void Parcel::addAddress() {
5     Address* newAddress = new Address();
6     newAddress->readIn();
7     stations.push_back( newAddress );
8   }
9
10  void Parcel::readIn() {
11    cout << "Bitte Trackingnummer eingeben: ";
12    cin >> trackingnumber;
13    cout << "Bitte Groessenklasse eingeben: ";
14    cin >> size;
15    cout << "Bitte Gewichtsklasse eingeben: ";
16    cin >> weight;
17    cout << endl << "Bitte Absender eingeben " << endl;
18    sender.readIn();
19    cout << endl << "Bitte Empfaenger eingeben " << endl;
20    destination.readIn();
21  }
22
23  void Parcel::outputDetails() {
24    cout << "Trackingnummer: " << trackingnumber << endl;
25    cout << "Groessenklasse: " << size << endl;
26    cout << "Gewichtsklasse: " << weight << endl;
27
28    cout << endl << "Absender : " << endl;
29    sender.printAddress();
30
31    cout << endl << "Empfaenger : " << endl;
32    destination.printAddress();
33
34    cout << endl << "Stationen : " << endl;
35    for( unsigned int i = 0; i < stations.size(); ++i ) {
36      if( stations[i] != NULL ) {
37        stations[i]->printAddress();
38        cout << endl;
39      }
40    }
41  }
42
43  Parcel::~Parcel() {
44    for( unsigned int i = 0; i < stations.size(); ++i ) {
45      delete stations[i];
46    }
```

```
47 }
```

parcelservice.h

```
1  #ifndef PARCELSERVICE_H
2  #define PARCELSERVICE_H
3
4  #include <string>
5  #include <vector>
6  #include "parcel.h"
7
8  using namespace std;
9
10 class ParcelService {
11 protected:
12   vector<Parcel*> parcels;
13 public:
14   void menu();
15   void addParcel();
16   void showAllParcels();
17   void searchParcel();
18   ~ParcelService();
19 };
20
21 #endif // PARCELSERVICE_H
```

parcelservice.cpp

```
1  #include <iostream>
2  #include "parcelservice.h"
3
4  void ParcelService::addParcel() {
5    Parcel* newParcel = new Parcel();
6    newParcel->readIn();
7
8    char antwort = ' ';
9    do {
10     cout << endl << "Moechten Sie eine (weitere) Station hinzufuegen? ↩
         (J)a/(N)ein ";
11     cin >> antwort;
12     antwort = toupper(antwort);
13     if( antwort == 'J' ) {
14       newParcel->addAddress();
15     }
16   } while( antwort == 'J' );
17
18   parcels.push_back( newParcel );
19 }
20
21 void ParcelService::showAllParcels() {
22   for( unsigned int i = 0; i < parcels.size(); i++ ) {
23     if(  parcels[i] != NULL ) {
24       cout << "PAKET " << i << endl;
25       parcels[i]->outputDetails();
26       cout << endl << endl;
27     }
28   }
29 }
```

```
30
31 void ParcelService::searchParcel() {
32   int number;
33   cout << "Bitte geben Sie die Trackingnummer ein: ";
34   cin >> number;
35   unsigned int i;
36   for( i = 0; i < parcels.size(); i++ ) {
37     if( parcels[i] != NULL && parcels[i]->getTrackingNumber() == ↩
         number ) {
38       cout << "PAKET " << i << endl;
39       parcels[i]->outputDetails();
40       cout << endl << endl;
41       break;
42     }
43   }
44   if( i < parcels.size() ) {
45     char antwort =  ' ';
46     do {
47       cout << endl << "Moechten Sie eine (weitere) Station hinzufuegen↩
            ? (J)a/(N)ein ";
48       cin >> antwort;
49       antwort = toupper(antwort);
50       if( antwort == 'J' ) {
51         parcels[i]->addAddress();
52       }
53     } while( antwort == 'J' );
54   } else {
55     cout << "Trackingnummer nicht gefunden" << endl;
56   }
57 }
58
59 void ParcelService::menu() {
60   int auswahl = 0;
61   while( true ) {
62     cout << endl << endl;
63     cout << "MENU " << endl;
64     cout << "(1) Paket hinzufuegen" << endl;
65     cout << "(2) Alle Pakete anzeigen" << endl;
66     cout << "(3) Pakete suchen" << endl;
67     cout << "(4) Exit" << endl;
68     cout << "Bitte waehlen: ";
69
70     cin >> auswahl;
71     switch( auswahl ) {
72     case 1: addParcel(); break;
73     case 2: showAllParcels(); break;
74     case 3: searchParcel(); break;
75     default: return;
76     }
77   };
78 }
79
80 ParcelService::~ParcelService() {
81   for( unsigned int i = 0; i < parcels.size(); i++ ) {
82     delete parcels[i];
83   }
84 }
```

parcelserviceV2.h

```
1  #include "parcelservice.h"
2
3  class ParcelServiceV2 : public ParcelService {
4  protected:
5    int findParcel();
6  public:
7    void menu();
8    void searchParcel();
9    void deleteParcel();
10 };
```

parcelserviceV2.cpp

```
1  #include <iostream>
2  #include "parcelserviceV2.h"
3
4  void ParcelServiceV2::menu() {
5    int auswahl = 0;
6    while( true ) {
7      cout << endl << endl;
8      cout << "MENU " << endl;
9      cout << "(1) Paket hinzufuegen" << endl;
10     cout << "(2) Alle Pakete anzeigen" << endl;
11     cout << "(3) Pakete suchen" << endl;
12     cout << "(4) Pakete loeschen" << endl;
13     cout << "(5) Exit" << endl;
14     cout << "Bitte waehlen: ";
15
16     cin >> auswahl;
17     switch( auswahl ) {
18     case 1: addParcel(); break;
19     case 2: showAllParcels(); break;
20     case 3: searchParcel(); break;
21     case 4: deleteParcel(); break;
22     default: return;
23     }
24   }
25 }
26
27 int ParcelServiceV2::findParcel() {
28   int number;
29   cout << "Bitte geben Sie die Trackingnummer ein: ";
30   cin >> number;
31   unsigned int i;
32   for( i = 0; i < parcels.size(); i++ ) {
33     if( parcels[i] != NULL && parcels[i]->getTrackingNumber() == ↩
         number ) {
34       return i;
35     }
36   }
37   return -1;
38 }
39
40 void ParcelServiceV2::searchParcel() {
41   int i = findParcel();
42   if( i < 0 ) {
43     cout << "Trackingnummer nicht gefunden" << endl;
```

```
44      return;
45    }
46    Parcel* parcel = parcels[i];
47    parcel->outputDetails();
48    cout << endl << endl;
49
50    char antwort =  ' ';
51    do {
52      cout << endl << "Moechten Sie eine (weitere) Station hinzufuegen? ↩
          (J)a/(N)ein ";
53      cin >> antwort;
54      antwort = toupper(antwort);
55      if( antwort == 'J' ) {
56        parcel->addAddress();
57      }
58    } while( antwort == 'J' );
59  }
60
61  void ParcelServiceV2::deleteParcel() {
62    int i = findParcel();
63    if( i < 0 ) {
64      cout << "Trackingnummer nicht gefunden" << endl;
65      return;
66    }
67    cout << "Paket mit Trackingnummer " << parcels[i]->getTrackingNumber↩
        () <<  " wird geloescht..." << endl;
68
69    delete parcels[i];
70    parcels.erase( parcels.begin() + i );
71  }
```