Prof. Dr. Thomas Breuel
Dipl.-Inf. Patrick Michel (AG ST)
Dipl.-Ing. Ludwig Schmidt-Hackenberg

**TU Kaiserslautern**
**Fachbereich Informatik**
**AG Bildverstehen und Mustererkennung**

# Übungsblatt 2: Software-Entwicklung 1 (WS 2010/11)

Ausgabe: in der Woche vom 01.11. bis zum 05.11.10
Abgabe: in der Woche vom 08.11. bis zum 12.11.10
Abnahme: max. zwei Tage nach der Übung

Bitte beachten Sie folgende Hinweise für dieses und die weiteren Übungsblätter:

1. Einzureichende Abgaben sind – sofern nicht explizit im Aufgabentext anders vermerkt – in *handschriftlicher Form* abzugeben. Hintergrund ist u.a. die Tatsache, dass Sie spätestens in der Abschlussklausur Aufgaben ebenfalls handschriftlich lösen müssen.
2. Vermeiden Sie den Einsatz integrierter Entwicklungsumgebungen wie beispielsweise Eclipse. Verwenden Sie den in der Vorlesung demonstrierten und auf dem ersten Übungsblatt eingeführten Editor GEdit. In der Klausur stehen ihnen diese Hilfsmittel schließlich nicht zur Verfügung.

## Aufgabe 1 Algorithmisches Beschreiben (Präsenzaufgabe)

Auf einem Stapel liegt ein Kartenspiel mit 36 Karten, die von 1 bis 36 nummeriert sind. Ein Roboter soll nun den Stapel bearbeiten. Der Roboter besitzt zwei Greifarme (im Folgenden als $L$ und $R$ bezeichnet) und kann drei Stapel (an Position $A$, $B$ und $C$) verwalten. Der Anfangs-Kartenstapel liegt ursprünglich unsortiert an Position $A$.

Der Roboter kann folgende einfache Aktionen ausführen:

- **Hole** oberste Karte mit Greifarm $X$ von Stapel $Y$, wobei $X \in \{L, R\}$ und $Y \in \{A, B, C\}$.
  Als Voraussetzung gilt, dass der Stapel nicht leer ist.
- **Lege** Karte von Greifarm $X$ auf Stapel $Y$, wobei $X \in \{L, R\}$ und $Y \in \{A, B, C\}$

Außerdem kann man den Roboter anweisen, Aktionen solange zu wiederholen, bis ein definierter Stapel leer ist:

- **Wiederhole** bis Stapel $Y$ leer ist, wobei $Y \in \{A, B, C\}$:
  - ... (Beliebige Folge von Aktionen)

Beispiel für eine Aktionsabfolge:

1. Hole oberste Karte mit Greifarm $L$ von Stapel $A$
2. Lege Karte von Greifarm $L$ auf Stapel $B$
3. Wiederhole bis Stapel $A$ leer ist
   a) Hole oberste Karte mit Greifarm $L$ von Stapel $A$
   b) Lege Karte von Greifarm $L$ auf Stapel $C$

Der Roboter hat mit dieser Aktionsfolge die oberste Karte des Anfang-Stapels $A$ nach $B$ verschoben und den restlichen Teil-Stapel in umgekehrter Reihenfolge an Position $C$ abgelegt.

*Hinweis:* Der Einfachheit halber können Sie Abkürzungen nach folgendem Muster verwenden:

- Hole oberste Karte mit Greifarm $L$ von Stapel $A$ $\Leftrightarrow$ **Hole** *L A*

*Hinweis: Zur Lösung dieser Teilaufgaben wird nur ein Greifarm benötigt.*

a) Teilen Sie den Anfangs-Kartenstapel (von $A$) in zwei gleich große Stapel (nach $B$ und $C$) auf.

b) Überführen Sie den Anfangs-Kartenstapel (von $A$) nach $B$, wobei wieder die ursprüngliche Reihenfolge der Karten gelten soll.

c) Überführen Sie den Anfangs-Kartenstapel (von $A$) nach $B$, wobei wieder die ursprüngliche Reihenfolge der Karten gelten soll. Die unterste Karte des Anfangsstapels soll jedoch auf Stapel $C$ liegen.

## Aufgabe 2 Algorithmisches Beschreiben (Einreichaufgabe)

Der Roboter kann nun folgende **zusätzliche** Aktionen ausführen:

1. **Vergleiche**, ob Wert der Karte in Greifarm $L$ kleiner als Wert der Karte in Greifarm $R$:
   - Wenn ja : ... (Beliebige Folge von Aktionen)
   - Wenn nein: ... (Beliebige Folge von Aktionen)
2. **Vertausche** Karten in beiden Greifarmen.

a) Suchen Sie aus dem Anfangs-Kartenstapel die größte Karte heraus, und legen Sie sie auf Stapel $B$ ab.

b) Suchen Sie aus dem Anfangs-Kartenstapel die größte Karte heraus, und legen Sie sie auf Stapel $B$ ab, wobei die restlichen Karten wieder an Position $A$ liegen sollen.

*Hinweis: Sie können frühere Aufgabenteile wieder verwenden, indem Sie Sequenzen von Operationen gruppieren und einen eindeutigen Namen zuordnen:*

z.B. $B$_NACH_$C$ =

1. **Hole** *L B*
2. **Lege** *L C*

c) Sortieren Sie den Anfangs-Kartenstapel nach absteigendem Wert und legen Sie ihn an Position $A$ ab.

*Hinweis: Sortieren Sie nach und nach die Karten mit größtem Wert aus und legen Sie sie auf Stapel B. Sorgen Sie dann dafür, dass der Stapel in der richtigen Richtung auf dem richtigen Stapel landet.*

d) *(freiwillige Zusatzaufgabe)* Suchen Sie aus dem Anfangs-Kartenstapel die größte Karte heraus, und legen Sie sie auf Stapel $B$ ab, wobei die restlichen Karten wieder *in der gleichen Ordnung* an Position $A$ liegen sollen.

## Aufgabe 3 Funktionen und Ausdrücke (Präsenzaufgabe)

| *Funktion* | *Beispiel* |
|---|---|
| $\text{inc}(x) = x + 1$ | $\text{inc}(1) = 2$ |
| $\text{triple}(x) = 3 * x$ | $\text{triple}(4) = 12$ |
| $\text{add}(x, y) = x + y$ | $\text{add}(1, 2) = 3$ |
| $\text{quadsum}(x, y) = \text{add}(x, y) * \text{add}(x, y)$ | $\text{quadsum}(2, 2) = 16$ |
| $\text{half}(x) = x \text{ div}^1 2$ | $\text{half}(8) = 4, \text{half}(11) = 5$ |

Für ganze Zahlen seien die obigen Funktionen definiert, mit deren Hilfe sich Ausdrücke formulieren lassen. Ein Beispiel für einen solchen Ausdruck ist `triple(inc(4))`. Einen Ausdruck auszuwerten bedeutet,

[1] div bezeichnet die ganzzahlige Division.

seinen Wert zu berechnen; so liefert `triple(inc(4))` den Wert `15`.

a) Beschreiben Sie in einem Satz, welche Berechnung jede der obigen Funktionen durchführt.

b) Welche Ergebnisse liefert die Auswertung folgender Ausdrücke?

| | |
|---|---|
| inc(99) | quadsum(inc(2), add(3, triple(1))) |
| add(13, 29) | half(inc(12)) |
| add(half(2), triple(2)) | triple(half(add(3, inc(3)))) |

c) Geben Sie drei verschiedene Ausdrücke an, deren Wert 37 ist. Jeder Ausdruck soll mindestens zwei Funktionen verwenden, und jede Funktion muss mindestens einmal vorkommen.

d) Schreiben Sie die nachfolgenden Funktionen (sie können +, −, ∗, div und die Funktionen von oben verwenden):

1. times5($x$): Multiplikation von x mit 5.
2. modulo($x, y$): Berechnet den Rest der ganzzahligen Division von x durch y.

## Aufgabe 4 Funktionen und Ausdrücke (Einreichaufgabe)

Im Folgenden werden Sie die Auswertung der Ausdrücke aus Aufgabe 3 dem Computer überlassen. Dazu wird der Haskell-Interpreter verwendet, den Sie auf dem vorherigen Übungsblatt kurz kennengelernt haben.

Damit der Interpreter die Ausdrücke auswerten kann, müssen Sie die verwendeten Funktionen definieren. In Haskell sehen die Definitionen der Funktionen wie folgt aus:

```
inc, triple :: Integer -> Integer
inc(x) = x + 1
triple(x) = 3 * x

add :: (Integer, Integer) -> Integer
add(x,y) = x + y
```

```
quadsum :: (Integer, Integer) -> Integer
quadsum(x,y) = add(x, y) * add(x, y)

half :: Integer -> Integer
half(x) = x `div` 2
```

*Hinweis:* Der Operator `div` muss in Haskell mit Hilfe von so genannten *Backticks* notiert werden. Dieses Zeichen liegt auf den meisten Tastaturen oben rechts und wird mit Hilfe der Umschalttaste erzeugt.

Die Definitionen werden in einer Datei gespeichert und dann vom Interpreter geladen (siehe Blatt 1, Aufgabe 2b). Um im Interpreter einen Ausdruck auszuwerten, geben Sie diesen hinter dem Interpreterprompt (nach dem '>'-Zeichen) ein.

Ein Beispiel für eine Sitzung mit dem Interpreter sehen Sie hier:

```
tux> ghci functions.hs
GHCi, version 6.10.4: http://www.haskell.org/ghc/  :? for help
Loading package ghc-prim ... linking ... done.
Loading package integer ... linking ... done.
Loading package base ... linking ... done.
[1 of 1] Compiling Main             ( functions.hs, interpreted )
Ok, modules loaded: Main.
*Main> triple(inc(4))
15
*Main> :browse Main
inc :: Integer -> Integer
triple :: Integer -> Integer
add :: (Integer, Integer) -> Integer
quadsum :: (Integer, Integer) -> Integer
half :: Integer -> Integer
*Main>
```

Zuerst definiert der Interpreter alle angegebenen Funktionen und lädt dafür die nötigen Module. Danach wird der Ausdruck `triple(inc(4))` ausgewertet. Der Interpreter liefert als Antwort den Wert des Ausdrucks. Mit dem `:browse` Kommando können Sie sich die Signatur aller definierten Funktionen ausgeben lassen.

a) Erstellen Sie eine Textdatei mit den Definitionen aller fünf Funktionen von oben und führen Sie diese Datei mit dem Interpreter aus.

*Hinweis: Wenn Sie Fehlermeldungen bekommen, überprüfen Sie ihre Datei, vermutlich hat sich ein Tippfehler eingeschlichen.*

b) Überprüfen Sie, ob ihre Ergebnisse aus den Aufgaben 3b) und 3c) korrekt sind.

c) Definieren Sie die Funktionen `times5` und `modulo`. Verwenden Sie die Datei aus Teil a) und starten Sie entweder den Interpreter mit der geänderten Datei neu oder erzwingen Sie ein erneutes Laden der Datei im Interpreter mit dem Kommando `:reload` oder kurz `:r`.

d) Schreiben Sie eine Funktion, welche die Fläche eines Rechtecks bestimmt. Benutzen Sie diese, um Funktionen zu schreiben, welche die Oberfläche von Quadern und Würfeln bestimmen.

e) Gegeben seien folgende zwei Funktionen:

```
f :: Int -> Int
f(0) = 1
f(n) = n * f(n-1)
```

```
g :: Integer -> Integer
g(0) = 1
g(n) = n * g(n-1)
```

1. Welchen Wert liefern die Ausdrücke `f(0)` bis `f(6)`? Versuchen Sie herauszufinden, was diese Funktionen berechnen.
2. Was passiert, wenn Sie `f` und `g` mit größeren Zahlen aufrufen? Können Sie sich vorstellen warum?

## Aufgabe 5 Grundkonzepte von Softwaresystemen

Kreuzen Sie an, ob folgende Aussagen wahr oder falsch sind. Bereiten Sie diese Aufgabe bis zu Ihrer nächsten Übungsstunde vor, so dass Sie bei Unklarheiten nachfragen und die Antworten diskutieren können.

| wahr | falsch | |
|---|---|---|
| ☐ | ☐ | Es gibt Softwaresysteme, die auf mehreren Rechnern verteilt laufen. |
| ☐ | ☐ | Entwurf und Programmierung stehen aufgrund ähnlicher Modelle und Konzepte zueinander in Beziehung. |
| ☐ | ☐ | Jede Programmiersprache besitzt eine formale Semantik. |
| ☐ | ☐ | Anforderungen werden immer in einer formalen Sprache beschrieben. |
| ☐ | ☐ | Da jede Programmiersprache ähnliche Konzepte mitbringt, fängt man beim Lernen einer neuen Programmiersprache nicht wieder bei Null an. |
| ☐ | ☐ | Für eine kontextfreie Grammatik $\Gamma = (N, T, \Pi, S)$ gilt $N \cap T = \{\}$ und $S \in N$. |
| ☐ | ☐ | Mit der Menge $\mathbb{T}_+ = \mathbb{N} \cup \{+\}$ als Alphabet kann die formale Sprache aller Terme der Form $a + b$, mit $a, b \in \mathbb{N}$ beschrieben werden. |
| ☐ | ☐ | Es gibt keine formale Sprache, die den Text dieses Übungsblatts beschreibt. |
| ☐ | ☐ | Das Startsymbol in einer Sprachdefinition mit Syntaxdiagrammen ist ein Terminalsymbol. |
| ☐ | ☐ | Auf der linken Seite einer Produktion in einer kontextfreien Grammatik steht genau ein Terminalsymbol. |
| ☐ | ☐ | Die Produktion $A \rightarrow abA$ kann nur an genau einer Stelle auf die Satzform $abA$ angewendet werden. |
| ☐ | ☐ | Die dadurch mögliche Ableitung ist eine Linksableitung. |
| ☐ | ☐ | Eine Grammatik, die keine Produktion mit mehr als einem Nichtterminalsymbol auf der rechten Seite besitzt, ist eindeutig. |
| ☐ | ☐ | Mit Syntaxdiagrammen lassen sich weniger Sprachen beschreiben als mit kontextfreien Grammatiken. |