

# Elektronische Prüfungen in der Informatik

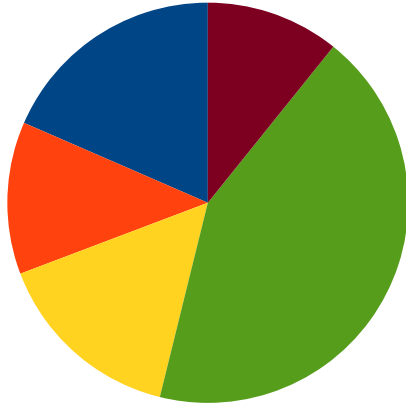
Ein Pilotversuch

---

Carsten Gips, FH Bielefeld ([carsten.gips@fh-bielefeld.de](mailto:carsten.gips@fh-bielefeld.de))

# **Ausgangssituation: Typisches Prüfungsszenario in der Informatik**

---



## Aufgabentypen

- Freie Aufgabe
- Multiple Choice
- Prgr: Ausgabe
- Prgr: Frei
- Prgr: Lücke

# Programmieren auf Papier ist praxisfremd

- Keine Unterstützung durch IDE, Compiler/Interpreter und andere Tools

```
TM1637.c (911af5d) ← TM1637.c (1771026)
GITLINS
▼ vorgaben master
  ▼ master ← origin/master
    Compare master (working) with
  ▼ Branches
  FILE HISTORY
  ▼ TM1637.c src/ledanzeige
    display_number(); remove 'cls'
    setup(); do not configure pins a
    brightness(); set global variable
    display(); send third command.
  LINE HISTORY
  ▼ TM1637.c #211 (911af5d) src/les
    initial upload: Vorgaben zu P1
  Compare -branch, tag, or ref- with
  Search by Message pattern or message
  or Author author: pattern or @pat
  or Commit ID commit: sha or # sha
  or Files file: glob or !glob
  or Changes change: pattern or --p
  176 --- /* (C90): Daten fuer Segment schicken */
  177 --- TM1637_start();
  178 --- TM1637_write_byte(seg | STARTADDR);
  179 --- TM1637_write_byte(seg_data);
  180 --- TM1637_stop();
  181
  182 --- /* (C90): Helligkeit setzen */
  183 --- TM1637_start();
  184 --- TM1637_write_byte(BRIGHT_OFFSET + bright);
  185 --- TM1637_stop();
  186 }
  187
  188 /*
  189 ** High-Level-Funktionen zum Setzen eines Segme
  190 ** Der Parameter number ist eine Gleitkommazahl
  191 ** Wird auf der Segmentanzeige mit drei Wörtern
  192 ** angezeigt.
  193 */
  194 void TM1637_display_number(float number) {
  195     TM1637_clear_display();
  196     TM1637_display(SEG1, TM1637_calculate_disp
  197     TM1637_display(SEG2, TM1637_calculate_disp
  198     TM1637_display(SEG3, TM1637_calculate_disp
  199     TM1637_display(SEG4, TM1637_calculate_disp
  200 }
  201
  202 /*
  203 ** Hilfsfunktion: Berechnet zu number die Blö
  204 ** Wird in TM1637_display_number() genutzt.
  205 */
  206 byte TM1637_calculate_display(float number, seg
  207     static int lastnumber = 0;
  208     static byte segmente[] = { 0x3f, 0x3f, 0x3f
  209     unsigned char digit;
  210     int remainder;
  211     /* Range-Check */
  212     if (number > 999.9f) {
  213         return 0;
  214     }
  215     while (number > 0) {
  216         digit = number % 10;
  217         remainder = number / 10;
  218         segmente[0] = (digit < 10 ? digit : 0);
  219         segmente[1] = (digit < 10 ? digit : 0);
  220         segmente[2] = (digit < 10 ? digit : 0);
  221         segmente[3] = (digit < 10 ? digit : 0);
  222         number = remainder;
  223     }
  224     return segmente[0];
  225 }
```

- Keine Hilfe aus dem Netz (Dokumentation, API, Stack Overflow, ...)

## Korrektur der Lösungen ist sehr aufwändig

- Handschrift (!)
- Syntaxfehler: Grenze zw. noch akzeptabel und fehlerhaft
- Oft „kreative“ Lösungsansätze: Manuelles Nachprüfen am Rechner ...

⇒ Typisch ca. 50..80 Teilnehmer mit ca. 15 Seiten pro Klausur

# **Pilotversuch: Elektronische Prüfungen mit ILIAS**

---

# Vorbereitung: Fragetypen im ILIAS

## Tests erstellen und verwalten

Das Test- und Assessment-System in ILIAS besteht technisch gesehen aus drei Grundkomponenten, die im Folgenden kurz erläutert werden sollen:

- **Fragen**

Die Fragen bilden die Grundlage für alle Tests, die mit dem Assessment-System vorgenommen werden können. Zu einer Frage können Sie einen Fragentext, Punktezahlen, mögliche Antworten, Bearbeitungszeiten etc. vergeben. Momentan sind im Test- und Assessment-System neun Fragetypen vorgesehen:

Auswahl	Eintragung	An- und Zuordnung	Markieren	Eigene Antwort verfassen	Sonstiges
<a href="#">Single-Choice-Fragen</a>	<a href="#">Lückentexte</a>	<a href="#">Anordnungsfragen</a> (vertikal und horizontal)	<a href="#">Fragentyp „Hotspot/Imageap“</a>	<a href="#">Freitext-Fragen</a>	<a href="#">Java-Applet-Fragen</a>
<a href="#">Multiple-Choice-Fragen</a>	<a href="#">Fragentyp „Numerische Antwort“</a>	<a href="#">Zuordnungsfragen</a>	<a href="#">Fragentyp „Fehler/Worte markieren“</a>	<a href="#">Datei-Upload-Fragen</a>	<a href="#">Flash-Applet-Fragen</a>
<a href="#">KPrim-Choice-Fragen</a>	<a href="#">Fragentyp „Begriffe benennen“</a>				
	<a href="#">Formelfragen</a>				

- **Fragenpools**

Fragenpools sind Container, die Sie zum Sammeln von Fragen verwenden **können**. Ob Sie einzelne Fragen nach inhaltlichen, zufälligen oder anderen Kriterien zu Fragenpools zuordnen, bleibt Ihnen vollständig überlassen (s. Kap. [Fragenpools erstellen und verwalten](#)).

# Vorbereitung: Übertragen der Aufgaben

Codeanalyse Haskell ID: [REDACTED]

Text einfügen

Betrachten Sie den folgenden Haskell-Code:

```
1 take 4 $ [(x,y) | x <- [1..], y <- [1..]]
2
3 take 4 $ [x+1 | x <- [1..], odd x]
4
5 take 4 $ [(n, even n) | n <- [1..]]
6
7 head (tail [1..4])
8
9 (head . tail) [1..4]
10
11 head . tail $ [1..4]
12
13 head . tail [1..4]
14
15 zip [1..2] ["one", "two", "three"]
16
17 [(x, y) | x <- [1..2], y <- ["one", "two", "three"]]
```

Geben Sie für alle Zeilen die Ergebnisse der Auswertung der jeweiligen Ausdrücke an.

- Zeile 1 (1P):
- Zeile 3 (1P):
- Zeile 5 (1P):
- Zeile 7 (0.5P):
- Zeile 9 (0.5P):
- Zeile 11 (0.5P):
- Zeile 13 (1P):
- Zeile 15 (1P):
- Zeile 17 (1.5P):

**Hinweis:** Wenn eine Zeile nicht ausgewertet werden kann, etwa durch Syntax- oder Typ-Fehler, geben Sie bitte als Antwort "Fehl" an.

Länge des Textfeldes

Werte \*

LÜCKE 2

Frage \*

Betrachten Sie den folgenden Haskell-Code:

```
1 take 4 $ [(x,y) | x <- [1..], y <- [1..]]
2
3 take 4 $ [x+1 | x <- [1..], odd x]
4
5 take 4 $ [(n, even n) | n <- [1..]]
6
7 head (tail [1..4])
8
9 (head . tail) [1..4]
10
11 head . tail $ [1..4]
12
13 head . tail [1..4]
14
15 zip [1..2] ["one", "two", "three"]
16
17 [(x, y) | x <- [1..2], y <- ["one", "two", "three"]]
```

Frage: p = img

Stunden: 0 Minuten: 8 Sekunden: 0

• Zeile 1 (1P): [gap 1] [(1,1), (1,2), (1,3), (1,4)], [(1,1), (1,2)], [(1,3), (1,4)], [(1,1), (1,2)], [(1,3), (1,4)] [gap]

• Zeile 3 (1P): [gap 2] [2, 4, 6, 8], [2, 4, 6, 8] [gap]

• Zeile 5 (1P): [gap 3] [(1, False), (2, True), (3, False), (4, True)], [(1, False), (2, True), (3, False), (4, True)], [(1, True), (2, True), (3, False), (4, True)] [gap]

• Zeile 7 (0.5P): [gap 4] [gap]

• Zeile 9 (0.5P): [gap 5] [gap]

Frage: p

Um eine Lücke in den Text einzufügen, setzen Sie den Cursor an die entsprechende Position und nutzen Sie die Schaltfläche "TextLücke". Anschließend stehen weiter unten entsprechende Bearbeitungsbereiche zur Verfügung. Ebenso können Sie die Lücken zur Bearbeitung im Lückentext direkt anklicken.

TextLücke

Zwischen Groß- und Kleinschreibung wird nicht unterschieden

Wenn Sie hier einen Wert eintragen, werden TextLücken, welche keinen eigenen Wert für eine maximale Länge definieren, sowie numerische Lücken mit dieser Länge erzeugt, so dass es nicht möglich ist eine größere Anzahl an Zeichen einzugeben. Für numerische Lücken ist darüber hinaus zu beachten, dass das Dezimaltrennzeichen dabei mit gezählt wird.

Wenn ausgewählt, werden Lücken mit identischen Lösungen entsprechend der Vorgaben bewertet, selbst wenn die gleiche Lösung mehrere Male verwendet wurde. Ist das Kontrollkästchen nicht ausgewählt, wird nur die erste verwendete identische Lösung bewertet.

TextLücke

80

Ist ein Wert größer 0 eingetragen, wird diese Lücke mit der hier angegebenen Länge erzeugt. Ist kein Wert angegeben, wird diese Lücke mit der global angegebenen Textfeldlänge erzeugt.

Antwort-Text	Punkte
[1,1], (1,2), (1,3), (1,4)	1 + -
[1,1], (1,2), (1,3), (1,4)	1 + -
[1,1], (1,2), (1,3), (1,4)	1 + -

Lücke erstellen



# Durchführung der E-Klausur

- Fragetypen-Demo für Teilnehmer (im Vorfeld)
- Teilnehmer in speziellen ILIAS-Kurs eintragen
- Test online schalten, wenn alle eingeloggt
- Bearbeitungszeit lässt sich auf Server begrenzen
  
- Ergebnis:
  - Ausdrucken der (teilautomatisch bewerteten) Klausur (PDF) plus Deckblatt
  - Teilnehmer unterschreiben ihre Klausur

# Korrektur und Bewertung

- MC vollautomatisch
- Lückentexte: teilautomatisch
- Freitext: manuelle Korrektur

ILIAS.FH Bielefeld - Klausur A <https://www.fh-bielefeld.de/elearning/ilias.php?r...>

9. Codeanalyse Haskell [10: 00:00]

Ihre Antwort:

Betrachten Sie den folgenden Haskell-Code:

```
1 take 4 $ [(x,y) | x <- [1..], y <- [1..]]
2
3 take 4 $ [x+1 | x <- [1..], odd x]
4
5 take 4 $ [(n, even n) | n <- [1..]]
6
7 head (tail [1..4])
8
9 (head . tail) [1..4]
10
11 head . tail $ [1..4]
12
13 head . tail [1..4]
14
15 zip [1..2] ["one", "two", "three"]
16
17 [(x, y) | x <- [1..2], y <- ["one", "two", "three"]]
```

Geben Sie für alle Zeilen die Ergebnisse der Auswertung der jeweiligen Ausdrücke an.

- Zeile 1 (1P): [(1,1),(1,2),(1,3),(1,4)] ✓
- Zeile 3 (1P): [2,4,6,8] ✓
- Zeile 5 (1P): [(1,false), (2, true), (3, false), (4, true)] ✗
- Zeile 7 (0.5P): 2 ✓
- Zeile 9 (0.5P): 2 ✓
- Zeile 11 (0.5P): 2 ✓
- Zeile 13 (1P): error ✗
- Zeile 15 (1P): [(1,"one"),(2,"two")] ✓
- Zeile 17 (1.5P): [(1,"one"),(1,"two"),(1,"three"),(2,"one"),(2,"two"),(2,"three")] ✓

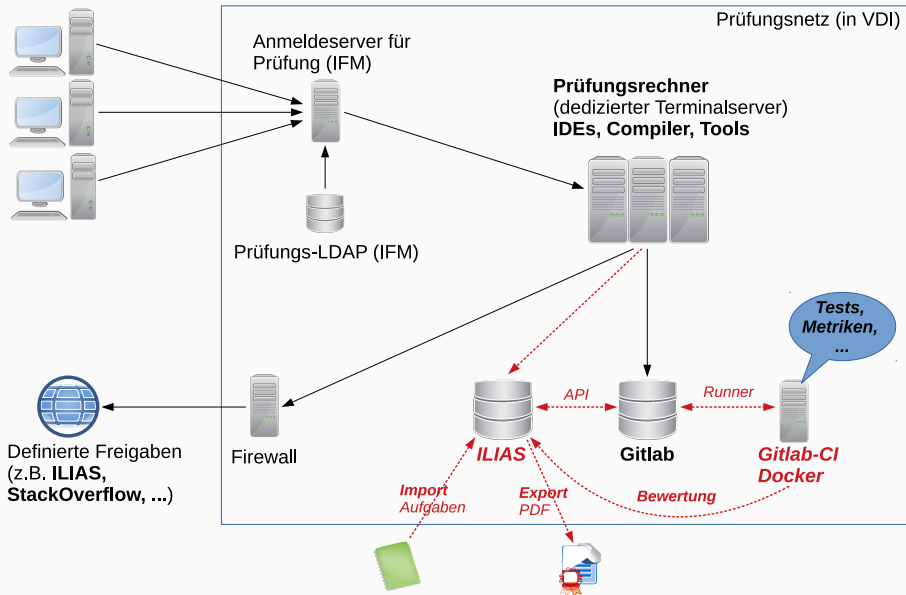
Hinweis: Wenn eine Zeile nicht ausgewertet werden kann, etwa durch Syntax- oder Typ-Fehler, geben Sie bitte als Antwort "Fehler" ein.

⇒ Korrektur auf Ausdruck oder im ILIAS

## **Zukunft: Konzeptskizze**

---

# Konzept für Prüfungsumgebung für die Informatik



**Vielen Dank! Fragen?**

---