Funktionale Programmierung

LVA 185.A03, VU 2.0, ECTS 3.0
WS 2019/2020

– Vorbesprechung –

(Stand: 30.09.2019)

Jens Knoop

Technische Universität Wien
Information Systems Engineering
Compilers and Languages

A Motivation

B Lernergebnisse

C Ablauf

D Ressourcen

Funktionale Programmierung im WS 2019/20

A Motivation

B L^3: Lernergebnisse, Lernaktivitäten, Leistungsnachweise

C Organisation, Ablauf

D Ressourcen

A

Motivation

A Motivation

B Lernergebnisse

C Ablauf

D Ressourcen

Funktionale Programmierung

...komplementiert und rundet die grundlegenden Lehrveranstaltungen zu wichtigen klassischen Programmierparadigmen und -stilen ab:

- **Funktionale Programmierung**
  LVA 185.A03 Funktionale Programmierung
  VU 2.0 ECTS 3.0
- Objektorientierte Programmierung
  LVA 185.A01 Objektorientierte Programmiertechniken
  VU 2.0 ECTS 3.0
- Logikorientierte Programmierung
  LVA 185.A12 Logikprogrammierung und Constraints
  VU 4.0 ECTS 6.0

...die alle auch in zugehörigen fortgeschrittenen Lehrveranstaltungen fortgeführt und vertieft werden.

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Generalthema der Lehrveranstaltung

...der **funktionale** Programmierstil, insbesondere:

- Programme als Systeme (wechselweise) rekursiver Rechenvorschriften.
- Lambda-Kalkül als rigorose semantische Fundierung funktionaler Programmiersprachen und Programmierung.
- Auswertungsordnungen für Ausdrücke und Programme, insbesondere sofortige und verzögerte Auswertung (engl. eager, lazy evaluation).
- Programmieren mit Funktionen als Argument und Resultat von Funktionen (sog. Funktionen höherer Ordnung).
- Polymorphie auf Datentypen und Funktionen.
- ...
- Umsetzung, Anwendung der Konzepte in Haskell.

A Motivation
B Lernergebnisse
C Ablauf
D Ressorucen

Im Mittelpunkt

"Can programming be liberated from the von Neumann style?"

John W. Backus (1924-2007)
Turing Award Preisträger 1977

John W. Backus. Can Programming be Liberated from the von Neumann Style? A Functional Style and its Algebra of Programs. Communications of the ACM 21(8):613-641, 1978. (Turing Award Speech)

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Lehrveranstaltungsgliederung (1)

- Teil I: Einführung
  - Warum fkt. Programmierung? Warum mit Haskell? Beispiele und Werkzeuge.
- Teil II: Grundlagen
  - Elementare Typen, Funktionen, Datentypdeklarationen.
- Teil III: Applikative Programmierung
  - Funktionen über elementaren Werten und Resultaten.
- Teil IV: Funktionale Programmierung
  - Funktionen über Funktionen, mit Funktionen als Argument und Resultat; Polymorphie auf Funktionen und Datentypen.

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Lehrveranstaltungsgliederung (2)

- Teil V: Fundierung funktionaler Programmierung
  - λ-Kalkül, Auswertungsordnungen, Typprüfung, Typinferenz.
- Teil VI: Weiterführende Konzepte
  - Ein-/Ausgabe, Fehlerbehandlung, Modulkonzept, Programmierprinzipien, Programmieren mit Strömen und Funktionen höherer Ordnung.
- Teil VII: Abschluss und Ausblick
  - Abschluss, Ausblick.
- Literaturverzeichnis
- Anhänge
  - Formale Rechenmodelle
  - Andere funktionale Sprachen

Zum Einstieg

...drei kurze süffig zu lesende Artikel aus den Jahren:

1998: Philip Wadler. Why no one uses Functional Languages. ACM SIGPLAN Notices 33(8):23-27, 1998.

...there is a tension between building useful systems and extending the frontiers of research ⟨*nothing less than a plea for functional programming*⟩.

2011: Yaron Minsky. OCaml for the Masses. Communications of the ACM 54(11):53-58, 2011.

...why the next language you learn should be functional!

2018: Neil Savage. Using Functions for Easier Programming. Communications of the ACM 61(5):29-30, 2018.

...when the limestone of imperative programming has worn away, the granite of functional programming will be revealed underneath.

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Warum die nächste Sprache

...funktional sein sollte:

- Konrad Hinsen. The Promises of Functional Programming. Computing in Science and Engineering 11(4): 86-90, 2009.

  ...adopting a functional programming style could make your programs more robust, more compact, and more easily parallelizable.

- Konstantin Läufer, Geoge K. Thiruvathukal. The Promises of Typed, Pure, and Lazy Functional Programming: Part II. Computing in Science and Engineering 11(5): 68-75, 2009.

  ...this second installment picks up where Konrad Hinsen's article "The Promises of Functional Programming" [...] left off, covering static type inference and lazy evaluation in functional programming languages.

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Functional Programming is Fun!

...weil funktionale Programmierung
etwas von der Eleganz der Mathematik
in die Programmierung bringt!

Peter Pepper. Funktionale Programmierung
in OPAL, ML, Haskell und Gofer.
Springer-V., 2. Auflage, 2003.

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Wo fkt. Progr. heute angewendet wird (1)

Von Wissenschaft:

- Jerzy Karczmarczuk. Scientific Computation and Functional Programming. Computing in Science and Engineering 1(3):64-72, 1999.

  ...modern functional programming languages and lazy functional techniques are useful for describing and implementing abstract mathematical objects in quantum mechanics.

- Noah M. Daniels, Andrew Gallant, Norman Ramsey. Experience Report: Haskell in Computational Biology. In Proc. 17th ACM SIGPLAN International Conference on Functional Programming (ICFP 2012), 227-234, 2012.

  ...Haskell gives computational biologists the flexibility and rapid prototyping of a scripting language, plus the performance of native code.

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Wo fkt. Progr. heute angewendet wird (2)

...über Wirtschaft:

- Curt J. Simpson. Experience Report: Haskell in the "Real World": Writing a Commercial Application in a Lazy Functional Language. In Proceedings of the 14th ACM SIGPLAN International Conference on Functional Programming (ICFP 2009), 185-190, 2009.

  ...describe[s] the initial attempt of experienced business software developers with minimal functional programming background to writ a non-trivial, business-critical application entirely in Haskell. ...discuss[es] the advantages and difficulties of Haskell in these circumstances, with a particular focus on issues that commercial developers find important but that may receive less attention from the academic community.

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Wo fkt. Progr. heute angewendet wird (3)

...bis System- und Web-Programmierung:

- Iavor S. Dachki, Thomas Hallgren, Mark P. Jones, Rebekah Leslie, Andrew Tolmach. Writing System Software in a Functional Language: An Experience Report. In Proceedings of the 4th International Workshop on Programming Languages and Operating Systems (PLOS 2007), Article No. 1, 5 Seiten, 2007.

  ...we describe our experience developing a prototype operating system, House, in which the kernel, device drivers, and even a simple GUI, are all written in Haskell.

- Michael Snoyman. Developing Web Applications with Haskell and Yesod. O'Reilly, 2012.

  ...supports high-performing applications that are modular, type-safe, and concise.

A Motivation
B Lernergebnisse
C Ablauf
D Ressorucen

Wo fkt. Progr. heute angewendet wird (4)

...ausgewählte sechs weitere Vorzeigebeispiele:

- Philip Wadler. An angry half-dozen. ACM SIGPLAN Notices 33(2):25-30, 1998.

  ...You've scrutinized functional languages. You've admired the elegance of lambda calculus, checked the benchmarks from the compilers, noted the security provided by strong typing. Now you want to know if they have been used to serious purpose. Mathematical elegance is well and good, but will it run that mission-critical system?

  Here are a half-dozen examplars of [serious] functional programs...

...insgesamt:

- Mihai Maruseac. Haskell: A Language for Modern Times. Crossroads, the ACM Magazine for Students 24(1):64-66, 2017.

Also lautet der Beschluss,
dass der Mensch was lernen muss.

Max und Moritz
Wilhelm Busch (1832-1908)
dt. Schriftsteller, satirischer Zeichner und Maler

B

L^3: Lernergebnisse, Lernaktivitäten, Leistungsnachweise

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Es war einmal: Lehrinhalte, Lehr- und Lernziele

...Ihnen das Rüstzeug für Ihren Wissens- und Fertigkeitserwerb in Theorie und Praxis an die Hand zu geben über

- die grundlegenden Konzepte und Prinzipien funktionaler Programmierung, ihrer theoretischen Fundierung, praktischen Umsetzung und Anwendung in funktionalen Programmiersprachen und funktionaler Programmierung.
- den angemessenen und zweckmäßigen Einsatz funktionaler Programmierung und ihrer Konzepte und Prinzipien für die Lösung programmiertechnischer Aufgaben.
- die konkrete Umsetzung und Anwendung dieser Konzepte und Prinzipien in der 'state-of-the-art' Programmiersprache Haskell, darunter auch Tipps, Tricks und mehr!

...die (positive) Beantwortung der Eingangsfrage: "Can programming be liberated from the von Neumann style?"

Vom Wissensvermittler_in zum Lernbegleiter_in

Change

früher		neu
- Input	**Orientierung** →	- Output
- Lehrinhalte	**Fokus** →	- Lernprozess
- Wissensvermittler_in	**Rolle der Lehrenden** →	- Lernbegleiter_in
- Konsument_innen	**Rolle der Studierenden** →	- Verantwortliche für Lernprozess
- gering	**Transparenz** →	- hoch

3

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

...mit höchster Veranlassung u. Unterstützung

3-Säulen der Umsetzung

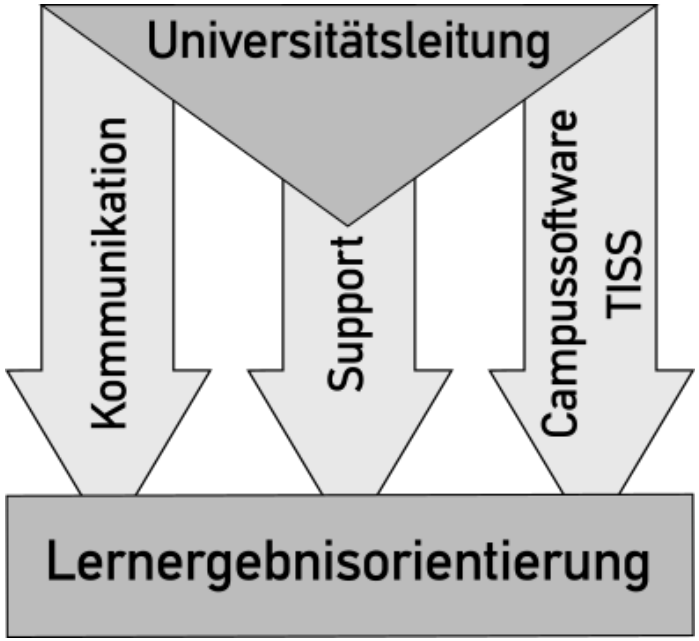

Abb.: 3-Säulen-Modell zur Einführung von Lernergebnisorientierung (TUW)

10

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Lernergebnisse

Nach positiver Absolvierung der LVA können Sie (u.a.)

1. die grundlegenden Prinzipien und Konzepte des fkt. Programmierstils und seiner theoretischen Fundierung
   - 1.1 erläutern und von anderen Programmierstilen wie prozeduraler, objektorientierter, logischer Prog. abgrenzen,
   - 1.2 am Bsp. von Haskell aufzeigen und illustrieren.
2. programmiertechnische Aufgabenstellungen im Kleinen
   - 2.1 in aufgabenangemessene Haskell-Programme umsetzen,
   - 2.2 aussagekräftige Testfälle für ihre Validierung erstellen,
   - 2.3 die Programme mit dem Interpretierer Hugs ausführen.
3. die Bedeutung u. Auswertungsw. v. Haskell-Programmen erklären und schrittweise mit Papier und Bleistift für verschiedene Auswertungsordnungen ausführen.
4. das gewählte Vorgehen sachlich und fachlich begründen.
5. Haskell-Prog. auf syn./sem. Korrektheit, Allgemeinheit, Wiederverw., Eff.-/Perf.-eigensch., angemessene Verw. fkt. progr.-sprachl. Konzepte untersuchen und beurteilen.

A Motivation
B Lernergebnisse
C Ablauf
D Ressorucen

Lernprozess: Methoden, Lernaktivitäten

...zum Erreichen der Lernergebnisse:

1. Angeleitetes eigenständiges Erlernen und Einüben: Durch Vorträge und umgekehrtes Klassenzimmer angeleitetes eigenständiges Erlernen und Einüben der in den Lernergebnissen beschriebenen Fähigkeiten mithilfe zur Verfügung gestellter Lehr- und Lernunterlagen, programmier- und nichtprogrammiertechnischer Übungsaufgaben und weiterer nach Bedarf selbstgewählter Materialien aus ergänzend und vertiefend vorgeschlagenen Lehrbüchern, Tutorien und wissenschaftlichen Originalarbeiten.
2. Vorbild- und rückmeldungsgeleitetes Lernen: Präsentieren, erläutern, begründen, vergleichen, wertend gegenüberstellen eigener und fremder Aufgabenlösungen aus sachl. und fachl. Sicht in tutorgeleiteten Übungsgruppen.
3. Selbsteinschätzungstests: Tests zur regelmäßigen Selbsteinschätzung und Selbstreflexion des eigenen bisherigen Lernfortschritts und Lernerfolgs.

Kompetenzbereiche (adressiert; nicht ausdrückl. unterrichtet)

Fachliche und methodische Kompetenzen

- Fachliche K.: Wissen um Grundlagen, Fundierung, Umsetzung und Anwendung des funktionalen Programmierstils.
- Methodische K.: Sach- und fachgerechte Anwendung programmiersprachlicher Konzepte des fkt. Programmierstils.

Kognitive und praktische Kompetenzen

- Kognitive K.: Abstraktes und mathematisch-logisches Denken, Analyse-, Verständnis- und Problemlösefähigkeit.
- Praktische K.: Angemessene Anwendung von Haskell auf konkrete programmiertechnische Aufgabenstellungen.

Soziale Kompetenzen und Selbstkompetenzen

- Soziale K.: Kommunikations-, Präsentations-, Argumentationsfähigkeit, Team- und Gruppenfähigkeit.
- Selbstk.: Selbstverantwortung, Selbstorganisation, Selbstreflexion, Wissbegierigkeit, Hartnäckigkeit, Ausdauer,...

Leistungsnachweise, Gesamtnote

Die Freude flieht auf allen Wegen,
der Ärger kommt uns gern entgegen.

Balduin Bählamm, der verhinderte Dichter
Wilhelm Busch (1832-1908)
dt. Schriftsteller, satirischer Zeichner und Maler

Leistungsnachweise

- Sieben beurteilte Abgaben programmiersprachl. Aufgaben.
- Eine beurteilte 90-minütige schriftliche Prüfung über Vorlesungs- und Übungsstoff.

Gesamtnote

Gewichtet im Verhältnis 1 zu 1 aus Beurteilungen der

- programmiersprachlichen Aufgabenlösungen
- schriftlichen Prüfung

falls beide Teilbeurteilungen positiv sind, sonst nicht genügend.

ECTS-Punkteaufschlüsselung

Angeleitete Lernaktivitäten	
– Vortrag (7 Einheiten $*$ 1.0h)	7.0h
– Umgek. Klassenzimmer (7 Einheiten $*$ 0.5h)	3.5h
– Übungsgruppen (7 Einheiten $*$ 1.5h)	10.5h
Eigenständige Lernaktivitäten	
– Selbstständiges Erarbeiten von Lernergebnissen (Vorschlag: I/3.0h, II/4.0h, III/4.0h, IV/6.0h, V/6.0h, VI/4.0h, VII/1.0h)	28.0h
– Speziell: Lösen der Übungsaufgaben (Vorschlag: 4 Ang. $*$ 2.5h $+$ 3 Ang. $*$ 4.0h)	22.0h
– Vorbereitung auf die schriftliche Prüfung	2.5h
Schriftliche Prüfung	1.5h
Summe	75.0h

Anmerkung: Die Angaben I bis VII beziehen sich auf Teil I bis Teil VII der Lehrveranstaltungsunterlagen.

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Wichtig

Programmieren ist wie schwimmen.
Man kann jahrelang zusehen,
ohne es zu lernen.

unbekannt

Der kürzeste Programmiererwitz:
Jetzt kann ich's.

unbekannt

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

C

Organisation, Ablauf

A Motivation

B Lernergebnisse

C Ablauf

D Ressourcen

Anmeldung zur Lehrveranstaltung

Anmeldung

- in TISS bis spätestens Freitag, 04.10.2019 (12:00 Uhr).

Voraussetzung einer validen Anmeldung:

- Erfolgreich abgeschlossene STEOP
  (Ausnahme: Erasmus$^+$-Studierende).

Studierende mit valider Anmeldung erhalten

- ein Benutzerkonto auf der Maschine
  `g0.complang.tuwien.ac.at`
- Benutzerkennung und erstes Losungswort per elektronischer Nachricht an ihre Standardadresse
  `e<Matr.Nr>@student.tuwien.ac.at`

zur Bearbeitung und Abgabe von Übungsaufgaben.

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Aufbau der LVA nach Lernaktivitäten

Fünf Hauptlernaktivitäten (neben weiteren eigenst. Lernakt.):

1. Angeleitete Lernaktivitäten
   - 1.1 Vortrag (7 Termine; wöchentlich/14-täglich dienstags, 08:15 - 09:15 Uhr).
   - 1.2 Umgekehrtes Klassenzimmer (7 Termine; wöchentlich/14-täglich dienstags, 09:15 - 09:45 Uhr).
   - 1.3 (Klein-) Gruppenübungen (7 Termine zu 90min.; wöchentlich, verschiedene Zeiten).
2. Programmiertechnische Aufgaben mit Einzelabgaben (7 Angaben, wöchentlich; beurteilt).
3. Nichtprogrammiertechnische Aufgaben (4 Angaben, wöchentlich; ohne Beurteilung).
4. Selbsteinschätzungstests als Teil der Ü-Gruppenarbeit (7 Tests, je 1 Test pro Ü-Gruppentermin; ohne Beurteilung).
5. Schriftliche Prüfung (90min., Do, 16.01.2020, 16:00 - 18:00 Uhr; beurteilt).

Übungsgruppenarbeit

1. Ausgabe von Aufgaben: Wöchentlich montags, abrufbar auf der Webseite der LVA; insgesamt 7 Angaben, davon die ersten 4 auch mit nichtprogrammiertechnischen Aufgaben, beginnend am Montag, den 07.10.2019.
2. Abgabe von prog.-techn. Lösungen (Erst-/Zweitabgabe):
   - 2.1 Erstabgabe: Eine Woche nach Ausgabe bis 12:00 Uhr; automatische Absammlung aus Home-Verzeichnissen (top-level, nicht in Unterverz.; Überprüfung mit Hugs!).
   - 2.2 Zweitabgabe: Nach Verbesserung der Erstabgabe (zur Abgabefrist siehe Dokument 'Allgemeine Hinweise zum Übungsablauf' auf LVA-Webseite).
   - 2.3 Erreichte Punkte: Hälfte der Summe beider Abgaben.
3. Nichtprogrammiertechnische Aufgaben: Keine Abgabe; Vorstellung, Besprechung und Diskussion von Lösungsvorschlägen durch Teilnehmer in den Übungsgruppen.

Übungsgruppenarbeit (fgs.)

4. Selbsteinschätzungstests: Je Übungsgruppentermin ein 15-minütiger Test; Besprechung und Diskussion von Lösungsvorschlägen durch Teilnehmer in den Übungsgruppen.
5. Übungsgruppenarbeit: Präsentieren, erläutern, begründen, vergleichen, wertend gegenüberstellen eigener und fremder programmier- und nichtprogrammiersprachlicher Aufgabenlösungen aus sachlicher und fachlicher Sicht.

Schriftliche Prüfung

- Dauer: 90 Minuten.
- Prüfungsstoff: Lehrveranstaltungsstoff, Übungsstoff, zwei zusammengehörende wissenschaftliche Artikel, die Sie sich im Lauf der Vorlesungszeit eigenständig erschließen (zugänglich aus TUW-Netz in IEEE Digital Library):
  1. Konrad Hinsen. The Promises of Functional Programming. Computing in Science and Engineering 11(4): 86-90, 2009.
  2. Konstantin Läufer, Geoge K. Thiruvathukal. The Promises of Typed, Pure, and Lazy Functional Programming: Part II. Computing in Science and Engineering 11(5): 68-75, 2009.
- Hilfsmittel: Keine.
- Anmeldung: Erforderlich (in TISS), Fristen beachten!

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Rückmeldungen zu Lernfortschritt, Lernerfolg

Drei Hauptformen:

1. Programmier- und nichtprogrammiertechnische Aufgaben
   - Vorstellung, Besprechung und Diskussion ausgewählter Lösungsvorschläge in den Übungsgruppen.
   - Direkte und indirekte Rückmeldung zu aktiver Beteiligung und Präsentationen in den Übungsgruppen.
   - Halbautomatische Testsystemrückmeldung zu Angaben.
   - Nach kapazitärer Möglichkeit: Tutorkommentierung einzelner Abgaben und Aufgaben (keine Notenrelevanz).
2. Selbsteinschätzungstests
   - Vorstellung, Besprechung und Diskussion der Lösungen in den Übungsgruppen.
   - Selbstreflexion.
3. Schriftliche Prüfung
   - Einsichtnahme nach abgeschlossener Verbesserung.

Gesamtbeurteilung, Zeugnisausstellung

- Positiver Abschluss: Nur, wenn praktische Übung (mindestens 50% der insgesamt möglichen Punkte) und schriftliche Prüfung (mindestens 50% der insgesamt möglichen Punkte) je für sich positiv abgeschlossen sind.
- Gesamtbeurteilung: Gewichtet im Verhältnis 1 zu 1 aus der Beurteilung der praktischen Übung und der schriftlichen Prüfung.
- Zeugnisausstellung: Nach dem ersten und jedem weiteren Antritt zur schriftlichen Prüfung; spätestens nach Ablauf des letzten Termins für die schriftliche Prüfung.

Voraussichtliche Termine

...für Vortrags-, umgek. Klassenz.-Einheiten, Angaben, Ü-Gruppen:

Vortrag, umgek. Klassenz.	Thema Vortrag	Thema umgek. Klassenz.
Di, 01.10.2019, 08:15-09:45	Teil I	n.a. / Vorbesprechung
Di, 08.10.2019, 08:15-09:45	Teil II	Teil I
Mo, 21.10.2019, 08:15-09:45	Teil III	Teil II
Di, 29.10.2019, 08:15-09:45	Teil IV	Teil III
Di, 12.11.2019, 08:15-09:45	Teil V	Teil IV
Di, 26.11.2019, 08:15-09:45	Teil VI	Teil V
Di, 10.12.2019, 08:15-09:45	Teil VII	Teil VI

Angabe	Ausgabe	Erstabgabe	Punkte	zugeh. ÜG-Termine
1	Mo, 07.10.19	Mo, 14.10.19	50	KW 43: 21.-25.10.19
2	Mo, 14.10.19	Mo, 21.10.19	50	KW 44: 28.-31.10.19
3	Mo, 21.10.19	Mo, 18.11.19	50	KW 45: 04.-08.11.19
4	Mo, 28.10.19	Mo, 04.11.19	50	KW 46: 11.-15.11.19
5	Mo, 11.11.19	Mo, 18.11.19	100	KW 48: 25.-29.11.19
6	Mo, 18.11.19	Mo, 25.11.19	100	KW 49: 02.-06.12.19
7	Mo, 25.11.19	Mo, 02.12.19	100	KW 50: 09.-13.12.19

Vorauss. Termine/Orte für die schriftl. Prüfung

Schriftl. Prüfung	Datum	Uhrzeit	Ort
Haupttermin	Do, 16.01.2020	16:00 - 18:00 Uhr	EI7/EI3
1. Nebentermin	Fr, 06.03.2020	15:00 - 17:00 Uhr	EI7
2. Nebentermin	Fr, 08.05.2020	15:00 - 17:00 Uhr	HS11
3. Nebentermin	Fr, 26.06.2020	15:00 - 17:00 Uhr	HS11

- Eine Anmeldung in TISS ist für Teilnahme an allen Terminen erforderlich (siehe TISS für An- und Abmeldefristen); merken Sie sich die Termine bitte vor und planen Sie entsprechend!
- Über die obigen Termine hinaus, keine weiteren Termine für die schriftliche Prüfung.
- Endgültige Termine, Zeiten, Orte, Fristen, Änderungen werden in TISS bekanntgegeben (im Zweifel gelten die Angaben in TISS).

D

Ressourcen

A Motivation

B Lernergebnisse

C Ablauf

D Ressourcen

Lehrbücher, wiss. Arbeiten, Haskell im Netz

...gezielte Hinweise auf

- Lehrbücher, wissenschaftliche Arbeiten
  - Siehe Vorlesungsunterlagen, detaillierte Leseempfehlungen am Ende jedes Kapitels.
- Haskell im Netz
  - Haskell-Homepage: `www.haskell.org/`
  - Haskell-Wiki: `wiki.haskell.org/Haskell/`
  - Haskell-Tutorial: `www.haskell.org/tutorial/`
  - Hugs-Interpretierer: `www.haskell.org/hugs`
- Haskell auf youtube
  - Interview mit John Hughes über 'Funktionale Programmierung und Haskell'.
    `https://www.youtube.com/watch?v=LnX3B9oaKzw`

Wichtige wiss. Zeitschriften und Konferenzen

...zur Publikation von Forschungsergebnissen im Umfeld funktionaler Programmierung und von Haskell sind besonders:

- Zeitschriftenreihe:
  - The Journal of Functional Programming. Matthias Felleisen, Jeremy Gibbons (Hrsg.), Cambridge, UK, seit 1991.
    `https://www.cambridge.org/jfp`
- Konferenz- und Symposiumsreihen:
  - ACM SIGPLAN International Conference Series on Functional Programming (ICFP), jährlich seit 1996.
    `http://www.sigplan.org/Conferences/ICFP`
  - ACM SIGPLAN International Haskell Symposium Series, jährlich seit 2008 (2002-2007 als ACM SIGPLAN Haskell Workshop Series).
    `https://wwww.haskell.org/haskell-symposium`

Arbeit mit TUW- und eigenen Rechnern

- Server für die praktischen Programmierübungen: `g0.complang.tuwien.ac.at`
- Arbeit mit TUW-Rechnern: Verfügbar im Labor Argentinierstraße 8, Erdgeschoss im Innenhof.
- Arbeit mit anderen, eigenen Rechnern: Z.B. zu Hause ist möglich.
- Abgaben von Übungsaufgaben: Ausschließlich auf `g0.complang.tuwien.ac.at`.
- Nötige Software: Hugs (frei verfügbar).
- **Wichtig:** Abgaben werden auf der g0 ausschließlich unter Hugs getestet. Überzeugen Sie sich deshalb stets von der gewünschten Funktionalität Ihrer Programmierlösungen auf der g0 unter Hugs!

Anlaufstellen

...bei Fragen und Problemen:

- Webseite der LVA:
  `www.complang.tuwien.ac.at/knoop/fp185A03.html`
- Umgekehrtes Klassenzimmer im Anschluss an den Vortragsteil
- Übungsgruppe

Vorlesungsmaterialien, Aufgaben, Termine

Denn was man schwarz auf weiß besitzt,
kann man getrost nach Hause tragen.

Faust. Eine Tragödie.
Johann Wolfgang von Goethe (1749-1832)
dt. Dichter und Naturforscher

- Webseite der Lehrveranstaltung:

`www.complang.tuwien.ac.at/knoop/fp185A03_ws1920.html`

Mitveranstalter, Tutoren

- Mitveranstalter:
  - Ass.Prof. Dipl.-Ing. Dr. techn. Ulrich Neumerkel
- Tutoren:
  - Lukas Grassauer, BSc
  - Niki Herl
  - Christoph Hochrainer, BSc
  - Samuel Pilz, BSc
  - Hannes Siebenhandl, BSc
  - Patrick Thier, BSc
  - Bruno Tiefengraber

Interesse an gefördertem Auslandsstudium?

Die Erasmus/LLP-Programmlinie der EU bietet eine Vielzahl lohnender Möglichkeiten, z.B.

- Linköping University, Schweden
- Aalto University, Finnland
- The University of Copenhagen, Dänemark
- Universität Halle-Wittenberg, Deutschland
- Universität Paderborn, Deutschland
- Universidad Politècnica de València, Spanien
- ...

Mehr dazu: `www.complang.tuwien.ac.at/knoop/erasmus`

A Motivation
B Lernergebnisse
C Ablauf
D Ressourcen

Zum Vorbesprechungsabschluss

Dabei sein ist
80 Prozent des Erfolges.

Woody Allen (* 1935)
amerik. Schauspieler und Regisseur

...wir, die FP-Teammitglieder, wünschen Ihnen viel (Lern-) Erfolg für diese Lehrveranstaltung und dass Sie von ihr profitieren, auch langfristig!

Nicht zuletzt:

Die Veranstaltung lebt mit Ihnen! Ihre Rückmeldungen, Anregungen, Verbesserungsvorschläge sind willkommen! Natürlich auch, wenn Ihnen etwas gut gefallen hat!