Fallbeispiel: Blended Learning Szenarien in einer einführenden Programmierausbildung

Prof. Dr. Detlef Krömker – Carsten Heep
Institut für Informatik
Johann Wolfgang Goethe-Universität

Übersicht

- Die Veranstaltung – Hintergründe
- Prinzipien der Umgestaltung
- Beispiele
- Ausblick

Randbedingungen – Grundlagen der Programmierung 1 (PRG-1)

Erste von drei Pflichtveranstaltungen zur Programmierung (250-350 Tn.)

- PRG 1 („Erste“ Sprache erlernen – Programmieren im „Kleinen“ - Grundkonzepte der praktischen Informatik) 9 CP
- PRG 2 (zweite + dritte Sprache – Konzepte) 8 CP
- PRG P (Praktikum – Programmieren im Team, Entwicklungsumgebungen – vierte Sprache) 8 CP
- Also ca. 16 % des Bachelorstudiengangs Informatik
  PRG 1 auch für Bioinformatik, Lehramt, alle Nebenfächler

Herausforderungen – PRG 1

- Einige Programmierprofis … viele (Studien-)Anfänger
- Fokus: Programmiersprache lernen … beherrschen (?)
- Viel Stoff: Die „Programmiersprache“ **+**
  - Grundbegriffe: Computer, Algorithmus, Programm , Daten, Information
  - Datentypen & Datenstrukturen
  - Prozedurales & Objektorientiertes Programmieren
  - Codierung und Diskretisierung
  - Algorithmen- und Softwareentwurf
- In einer klassischen 4V+2Ü Veranstaltung lernten mehr als „50%“ das Programmieren NICHT! – haben sich „durchgemogelt“

Umstellungskonzept (erstmals WS 09/10)

- 4V + 2Ü 2V + **2E** + 2Ü also ein Blended Learning Konzept

- **2E**: LernBar eLearning Kurse: erscheinen wöchentlich, mind. 3 p.W.
  - fachliche Vertiefung mit Eingangs- und Abschlussfragen

  - Programmiersprachendetails und viele, sehr kleine Programmierbeispiele: lauffähig, änderbar, …

  -

  - **Nachbereitung der Vorlesung – typisch 8 Fragen**
    Nachlesen (und Verstehen) im Skript oder eLecture noch einmal anschauen

2Ü + 2V

Hausübung in Tutorengruppe, wöchentlich

- strukturiert und gut vorbereitet: Tutoren sollen aktivieren!
- viele Programmieraufgaben, kleine und

- 3 große Programmieraufgaben (14-tägig)
  ca. 200-300 Zeilen Code, dort Peer-Programming (2 Studierende) …

- Ergebnisse schnell und immer in Moodle verfügbar!

- Zwischentest nach 7 Wochen (vor Weihnachten – wg. Weihnachtskrise))

Vorlesung: ganz klassisch: NUR Konzepte, Übersicht, Motivation, …

Blended Learning in PRG-1

Prof. Dr. Detlef Krömker – Carsten Heep
Institut für Informatik

25. Oktober - 31. Oktober

Diese Woche finden zwei Vorlesungen statt; Mittwoch (27.10.2010) 14-16 und Freitag (29.10.2010) 10-12 Uhr.

Skript (V02-Programmieren - Erste Schritte, V03 - Elementare Datentypen)

Folien (Vorlesung 2, Vorlesung 3)

E-Kurs (Grundkonzept Imperativer Sprachen, Zahlendarstellung, Elementare Datentypen *Quiz*)

Grundkonzept Imperativer Sprachen, Zahlendarstellung

Lösung zu Aufgabenblatt Nr.01, Lösung zu Aufgabenblatt Nr.01

Programmieren - Erste Schritte

Übungsblatt Nr. 01

Abgabe zum Übungsblatt Nr. 01

1. November - 7. November

Skript (V4 Kontrollstrukturen)

Folien (4. Vorlesung)

E-Kurs (Kontrollstrukturen, Strukturiertes Programmieren, Grundlagen (Quiz))

Kontrollstrukturen, Strukturiertes Programmieren, Go To Statement Considered Harmful

Lösungen zu Aufgabenblatt Nr.02, Lösungen zu Aufgabenblatt Nr.02, Lösungen zu Aufgabenblatt Nr.02

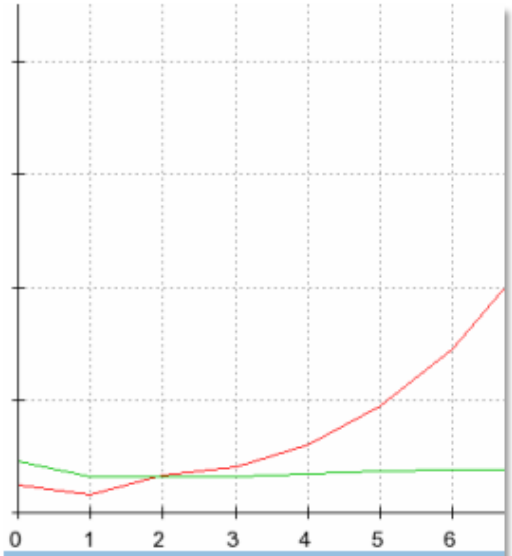
Kontrollstrukturen

Übungsblatt Nr. 02

Abgabe zum Übungsblatt Nr. 02

Rekursion

Die rekursive Variante benötigt bei ihrer Ausführung linear viel Speicher da für jeden Funktionsaufruf auf dem Systemstapel Speicher zum Verarbeiten der Funktion allokiert wird. Die iterative Version hingegen benötigt nur einen konstanten Speicherplatz für die drei Variablen **n**, **f** und **i**. Die Anzahl der Rekursionsaufrufe oder der rekursiven Schritte bezeichnen wir als *Rekursionstiefe*. Für die Fakultät einer Zahl *n* liegt sie nach dem oben gegebenem Codebeispiel bei *n–1*, da wir *n–1* mal die Funktion **factorial** aufrufen. In der Abbildung sehen wir die Abhängigkeit zwischen Laufzeitzeit und der Größe des Argumentes *n* zwischen rekursiver und iterativer Variante grafisch dargestellt.

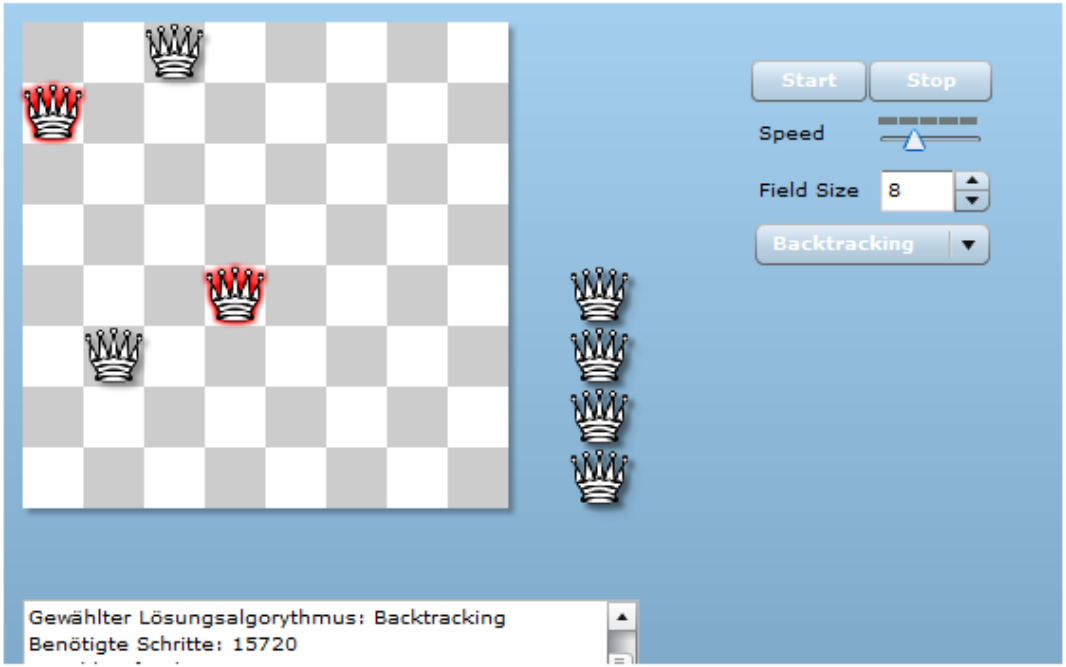


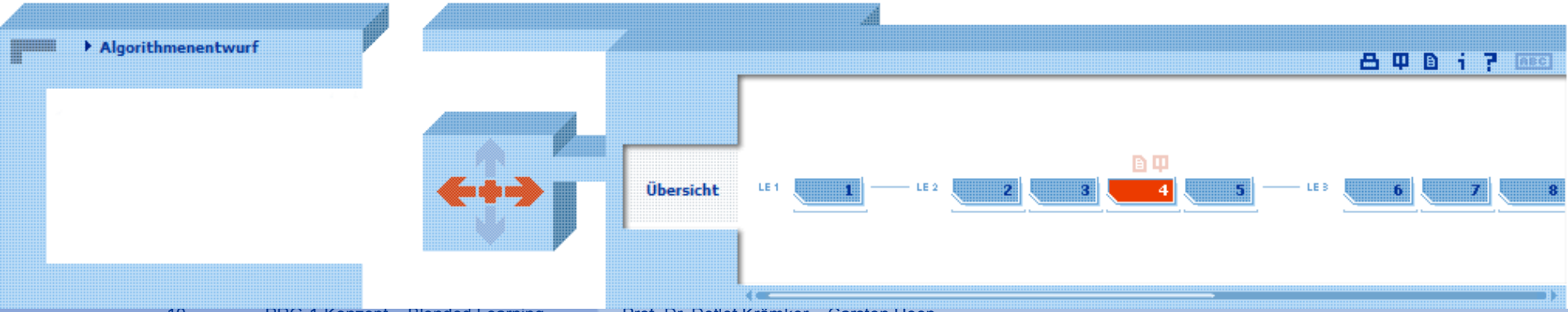
Laufzeitverhalten für die Rekursive und Iterative Variante zu Berechnung der n'ten Fibonacci-Zahl

Speicheranforderungen und Laufzeit scheinen gegen die Verwendung von Rekursionen zu sprechen. Allgemein lässt sich jedoch festhalten, dass es gewisse Problemstellungen gibt, bei denen es einfacher ist, eine rekursive Lösung zu finden obwohl *rekursion* und *iteration* gleichmächtig sind. Hinter der Rekursion steht der Gedanke des Zerteilens eines Problem in ein oder mehrere leichtere Probleme gleichen Typs. Das klassische Beispiel, bei dem es einfacher ist eine Rekursive Lösung zu finden, sind die sogenannten *Türme von Hanoi*.

Das N-Damen-Problem

Positionieren Sie die Damen so auf dem Spielfeld, das diese sich weder horizontal, vertikal noch diagonal schlagen können. Konflikte werden durch eine rote Umrandung der Dame signalisiert.





Das N-Damen-Problem

Dieser Algorithmus ist zwar intuitiv, aber er garantiert uns nur ein lokales Minimum: Was machen wir, wenn am Ende noch Konflikte existieren und keine Verschiebung irgendeiner Dame die Konfliktzahl mehr mindern kann? In diesem Fall sind wir mit unserem Latein am Ende: Wir setzten die Damen noch einmal neu in Zufallspositionen und beginnen von vorn.

Es gibt noch viele weitere Beispiele für Greedy-Algorithmen, etwa die Suche nach minimalen Spannbäumen in Graphen, oder die Berechnung von Wechselgeld mit einer möglichst geringen Anazahl an Münzen die wir hier nicht behandeln wollen.

```
27 class QueenGreedy(QueensProblem):
28     def solve(self):
29         self.Solution = [randint(0, self.SIZE-1) for i in range(self.SIZE
30 
31         while True:
32             self.print_board()
33             q = self.get_queen()
34             if q==None:
35                 print "Finish!"
36                 break
37             else:
38                 print "%i attacks" % q
```

```
Jython 2.5.1 (Release_2_5_1:6813, Sep 26 2009, 13:47:54)
[Java HotSpot(TM) Client VM (Sun Microsystems Inc.)] on java1.6.0_21


>>>
```

IDLE | EXEC | STOP | RESET | LOAD | SAVE | CHECK

Blended Learning in PRG-1

„Seiteneffekte“ durch Einsatz von E-Kursen:

- Wiederholung zur Prüfungsvorbereitung
- Keine zeitliche Bindung für Studierende
- Erleichterter „Quereinstieg“ (Nebenfächler, fachfremde M.Sc.)
- Förderung des Selbststudiums

Blended Learning in PRG-1

Was noch? - Moodle!

- Bereitstellung der Materialien (Zentralität)
  - *Skript, Folien, eKurse, Readings, Übungsblätter und eLectures*
- Online Abgabe der Übungen
- Informationsverteiler
- Forumsdiskussionen
- **WS 10/11 zusätzlich Mentoring**

Fazit

Fazit für PRG-1

- Quote erfolgreiche Teilnehmer von PRG-1 lag bei 80% im WS 2009/2010
- Mehr Flexibilität im Studium
- Direkte Vermittlung des eignen Wissensstands (Feedbacks)
- Besserer Überblick des eigenen Leistungsstands moodle (Bewertungssystem)
- **Sehr gutes Feedback von den Studierenden**

Ausblick

Reakkreditierung 2011 (nach 5 Jahren)

- PRG 1 hat dann 11 CP, also + 2 CP
  - mehr (oder richtigerweise besser bemessener) Übungsanteil
  - sehr wenig mehr Stoff

- „Studieren lernen“, als Pflichtfach + Coaching Mentoring (2 CPs)

- eKurse weiter verbessern!

- „Übergangskurse“: - Umstieg auf eine andere Programmiersprache
  - PYTHON JAVA wie macht man das?
  - Entwicklungsumgebung
  - - GUI – Entwicklung
  - - Netzwerkprogrammierung

Prof. Detlef Krömker, Carsten Heep

Professur für Graphische Datenverarbeitung
Institut für Informatik
Fachbereich Informatik und Mathematik
Goethe-Universität Frankfurt/Main

kroemker@gdv.cs.uni-frankfurt.de
cheep@gdv.informatik.uni-frankfurt.de

www.gdv.informatik.uni-frankfurt.de