Motivation

Warum...

- funktionale Programmierung?
- Haskell?

Warum funktionale Programmierung?

Programmierparadigmen...

- imperativ
  - prozedural (Pascal, Modula, C,...)
  - objektorientiert (Smalltalk, Oberon, C++, Java,...)
- deklarativ
  - funktional (Lisp, ML, Miranda, Haskell, Gofer,...)
  - logisch (Prolog und Varianten)
- ...

Ein Vergleich - imperativ vs. funktional

Gegeben ein Problem P.

Imperativ: Typischer Lösungsablauf in 3 Schritten

1. Beschreibe eine(n) Lösung(sweg) L für P
2. Gieße L in die Form einer Menge von Anweisungen für den Rechner
3. Organisiere die Speicherverwaltung

Ein Beispiel

Problem: Bestimme die Komponentenwerte eines Feldes ganzer Zahlen mit Wert kleiner oder gleich 10

Typische Lösung imperativ...

```
...
j := 1;
FOR i:=1 TO maxLength DO
    IF a[i] <= 10 THEN b[j] := a[i]; j := j+1 FI
OD;
...
```

Stichwort: **Softwarekrise!**

Zum Vergleich...

Essenz funktionaler Programmierung: Statt des “wie” das “was” in den Vordergrund stellen!

Typische Lösung funktional...

```
...
b :: [Int]
b = [ n | n <- a, n <= 10 ]
...
```

Stichwort: **list comprehension!**

...typisch und spezifisch für funktionale Sprachen!

Zum Vergleich: $\{n \mid n \in a \wedge n \leq 10\}$

...etwas der Eleganz der Mathematik in die Programmierung!

Vorteile funktionaler Programmierung

- Einfach(er) zu erlernen
  ...da weniger Grundkonzepte (insbesondere: keine Zuweisung, keine Schleifen, keine Sprünge)
- Höhere Produktivität
  ...da Programme dramatisch kürzer als funktional vergleichbare imperative Programme (Faktor 5 bis 10)
- Höhere Zuverlässigkeit
  ...da Korrektheitsüberlegungen/-beweise einfach(er) (math. Hintergrund, keine durchscheinende Maschine)

Nachteile funktionaler Programmierung

- Geringe(re) Effizienz
  ...aber: enorme Fortschritte, zudem Korrektheit vorrangig gegenüber Geschwindigkeit, weiters einfache(re) Parallelisierbarkeit
- Gelegentlich unangemessen, etwa für inhärent zustandsbasierte Anwendungen oder zur GUI-Programmierung
  ...aber: Anwendungseignung ist stets zu überprüfen; kein spezifisches Problem fkt. Programmierung

Warum Haskell?

Ein Blick auf andere funktionale (Programmier-)sprachen...

- λ-Kalkül (Ende der 30er-Jahre, Alonzo Church, Stephen Kleene)
- Lisp (frühe 60er-Jahre, John McCarthy)
- ML, SML (Mitte 70er-Jahre, Michael Gordon, Robin Milner)
- Hope (um 1980, Rod Burstall, David McQueen)
- Miranda (um 1980, David Turner)
- OPAL (Mitte der 80er-Jahre, Peter Pepper et al.)
- Haskell (Ende der 80er-Jahre, Paul Hudak, Philip Wadler et al.)
- Gofer (Anfang der 90er-Jahre, Mark Jones)
- ...

Warum also nicht Haskell?

Haskell ist...

- eine moderne funktionale Sprache
  - starke Typisierung
  - lazy evaluation
  - pattern matching
  - Polymorphie
  - Generizität
  - Datenabstraktion (abstrakte Datentypen)
  - Modularisierung (Programmierung im Großen)
  - ...
- eine Sprache für "real world" Probleme
  (s.a. http://www.cs.bell-labs.com/~Swadler/realworld/)
  - mächtige Bibliotheken
  - Schnittstellen z.B. zu C
  - ...

Nicht zuletzt: Wenn auch komplex, ist Haskell eine "gute" Lehrsprache, auch dank Hugs!

Fassen wir noch einmal zusammen...

Steckbrief: **Funktionale Programmierung**

Grundlage:	Lambda-Kalkül
Abstraktion:	Funktionen (höherer Ordnung)
Eigenschaft:	referentielle Transparenz
historische Bedeutung:	Basis vieler Programmiersprachen
Anwendungsbereiche:	Theoretische Informatik Artificial Intelligence experimentelle Software Programmierunterricht
Programmiersprachen:	Lisp, ML, Miranda, Haskell

sowie...

Steckbrief: **Haskell**

benannt nach:	Haskell B. Curry (1900-1982) http://www-gap.dcs.st-and.ac.uk/~history/Mathematicians/Curry.html
Paradigma:	rein funktionale Programmierung
Eigenschaften:	lazy evaluation, pattern matching
Typsicherheit:	stark typisiert, Typinferenz modernes polymorphes Typsystem
Syntax:	komprimiert, intuitiv
Informationen:	http://haskell.org http://haskell.org/tutorial/
Interpretierer:	Hugs (http://haskell.org/hugs/)

Ein erstes Haskell-Programm...

```
{-######################################################
   FirstScript.hs  ...''traditional scripts'' erhalten
                    die Dateiendung .hs
######################################################-}

-- Die konstante Funktion sum17and4
sum17and4 :: Int
sum17and4 = 17+4

-- Die Funktion square zur Quadrierung einer ganzen Zahl
square :: Int -> Int
square n = n*n

-- Die Funktion double zur Verdopplung einer ganzen Zahl
double :: Int -> Int
double n = 2*n

-- Die Funktion doubleSquare, eine Anwendung der vorherigen
doubleSquare :: Int
doubleSquare = double (square (4711 - sum17and4))
```


...in alternativer Darstellung.

```
#######################################################
   FirstLiterate.lhs  ...''literate scripts'' erhalten
                      die Dateiendung .lhs
#######################################################

Die konstante Funktion sum17and4
>     sum17and4 :: Int
>     sum17and4 = 17+4

Die Funktion square zur Quadrierung einer ganzen Zahl
>     square :: Int -> Int
>     square = n*n

Die Funktion double zur Verdopplung einer ganzen Zahl
>     double :: Int -> Int
>     double = 2*n

Die Funktion doubleSquare, eine Anwendung der vorherigen
>     doubleSquare :: Int
>     doubleSquare = double (square (4711 - sum17and4))
```


Zum ersten Aufgabenblatt...

- ...ein erweitertes Aufgabenblatt
- ...erhältlich nur im Web unter folgender URL
  http://www.complang.tuwien.ac.at/knoop/lehre/fktPrg-lu185161.html ...in Kürze
- Abgabe: Di, den 04.11.2003, 12:00 Uhr
- "Nachträgliche" Abgabe: Di, den 11.11.2003, 12:00 Uhr

Vorschau:
Ausgabe des zweiten Aufgabenblatts: Di, den 04.11.2003

Vorschau auf die nächsten Vorlesungstermine...

- **Heute**, Di, 14.10.2003, 18-19 Uhr, HS1 im Freihaus, 1. Zusatzvorlesung
- Di, 21.10.2003 und Di, 28.10.2003 keine Vorlesung
- Ab Di, 04.11.2003, jeden Dienstag von 13-14 Uhr im Treitl-Hörsaal
- **Zusätzlicher Termin**: Di, 11.11.2003, 18-19 Uhr, Treitl-Hörsaal, 2. Zusatzvorlesung