# Deklarative Programmierung
# WS 22/23

Michael Hanus

11. Januar 2023

## Detaillierter Vorlesungsverlauf

**24.10.** Organsiatorisches;
**Einführung in die funktionale Programmierung:** Variablenbegriff, Programm, Funktionsdefinitionen, Ausdrücke, Beispiel `square`, Beispiele `min`/`fac`, Auswertungsmöglichkeiten, Fibonacci-Zahlen (rekursiv)

**26.10.:** Fibonacci-Zahlen (iterativ) Lokale Definitionen, Layout-Regel, Vorteile lokaler Definitionen; Basisdatentypen, Typannotationen, algebraische Datentypen (Aufzählungstypen, Verbundtypen, gemischte Typen, Listen), Ausgabe von Daten (`deriving Show`)

**2.11.** Operatoren, polymorphe Funktionen (`length`, `(++)`, `last`), Definition polymorpher algebraischer Datentypen, `Maybe` und `maybeHead`, Binärbäume, `String`, Vereinigungstypen (`Either`), Tupel (`fst`, `snd`), Funktionen `zip`

**7.11.** Funktion `unzip`, Testen mit QuickCheck: Eigenschaften, `==>`, Referenzimplementierung, Regressionstests, automatisierte Ausführung aller Tests, Tests mit polymorphen Funktionen, Pattern Matching (Patternaufbau, case-Ausdrücke), Guards

**9.11.** Funktionen höherer Ordnung, anynome Funktionen, partielle Applikation, Currying, Sections, Funktion `flip`, generische Programmierung (`map`, `foldr`, `filter`, `foldl`)

**14.11.** Kontrollstrukturen als Funktionen höherer Ordnung (`while`), Funktionen als Datenstrukturen (Implementierung von Feldern), wichtige Funktionen höherer Ordnung (Komposition, `curry/uncurry`, `const`), Funktionen höherer Ordnung in imperativen Sprachen (Python, Java 8, JavaScript)

**16.11.** Motivation und Struktur von Typklassen, Instanzen, vordefinierte Funktionen in Typklassen, Typklassen für polymorphe Datentypen, Standardklassen, `deriving`, Klasse `Read` und Funktionen `read` und `reads`; Lazy Evaluation, Rechnen mit unendlichen Datenstrukturen (`from`, `primes`)

**21.11.** Rechnen mit unendlichen Datenstrukturen (`fibs`, `repeat`, `iterate`), Lazy Evaluation, Sharing, Graphreduktion, Vorteile von Lazy Evaluation, arithmetische und andere Sequenzen, Typklassen `Enum` und `Bounded`, List comprehensions

**23.11.** Idee der Ein-/Ausgabe, I/O-Aktionen, Aktionen zur Ein- und Ausgabe, do-Notation, Beispiel Ausgabe von Zwischenergebnissen, I/O-Aktionen zum Lesen und Schreiben von Dateien, Zeilen einer Datei numeriert ausgeben

**28.11.** Module, Exportdeklarationen, Importdeklarationen, Erzeugung von ausführbaren Programmen mittels `ghc`; Transformationen auf Containerstrukturen: `Functor`, `fmap`, `Functor`-Gesetze

**30.11.** Transformationen mit beliebigen Funktionen: `pure`, `<*>`, `Applicative`, arithmetische Ausdrücke und deren Auswertung, `Applicative`-Gesetze, `Applicative`-Instanzen für Listen und `IO`, effektvolle Berechnungen, Verbesserung der Auswertung arithmetischer Ausdrücke durch monadische Struktur, Klasse `Monad`

**5.12.** `Monad`-Instanz für Listen, Monadengesetze; QuickCheck: Eingabeklassifikation mit `classify` und `collect`, eigene Definitionen von Testdaten: Klasse `Arbitrary`, `elements`, `choose`, `oneof`, `sized`, `vector`, Fallstudien Peano-Arithmetik, `frequency`

**7.12. Einführung in die Logikprogrammierung:** Motivation, Verwandtschaftsbeispiel, Prolog-Programme, Fakten, Regeln, Anfragen, Prolog-Syntax (Zahlen, Atome, Strukturen, Listen), Variablen, Rechnen mit Listenstrukturen Operatoren, Gleichheit von Termen

**12.12.** Programmiertechnik Aufzählung des Suchraumes (Färben einer Landkarte, Sortieren von Zahlenlisten), Programmiertechnik Musterorientierte Wissensrepräsentation (`append`), Verwendung von Relationen, Peano-Zahlen (Definition, Addition, Subtraktion)

**14.12.** Rechnen in der Logikprogrammierung: einfaches Resolutionsprinzip, Substitution, Unifikator, mgu, disagreement sets, Unifikationsalgorithmus, occur check, Komplexität, allgemeines Resolutionsprinzip (SLD-Resolution)

**19.12.** Auswertungsstrategie und SLD-Baum, Beweisstrategie von Prolog, Endlosschleifen, Negation als Fehlschlag, Probleme der Negation, verzögerte Negation mit Coroutining; Cut-Operator, Fallunterscheidung

**21.12.** Arithmetik in Prolog (`is`, Fakultätsfunktion), arithmetische Constraints, Beispiel Schaltkreisanalyse, Beispiel Hypothekenberechnung, Constraint-Programmierung über endlichen Bereichen, allgemeines Vorgehen, send-more-money-Beispiel, 8-Damen-Problem, verbesserte Labeling-Strategien

**9.1.** Meta-Programmierung: Prädikate höherer Ordnung (`call`, `maplist`), Kapselung des Nichtdeterminismus (`findall`, `bagof`, `setof`), Veränderung der Wissensbasis (`assert`, `retract`), Meta-Interpretierer zur Beweislängenberechnung, Prädikate zur Ein- und Ausgabe von Daten

**11.1.** Tracing von Prolog-Programmen, Differenzlisten, Definite Clause Grammars, Kurzüberblick zu Multiparadigmen-Sprachen, Einführung in Curry (Verwandtschaftsbeispiel), funktionale Muster (`last`, `perm`), bedarfsgesteuerte Suche in Curry