Informatik

Programmiersprachen

Programmiersprachen

- Künstliche Sprache, exakt definiert
- Ziel: Geräte steuern
- Quelle: Menschen oder Maschinen (z.B. pdf)
- Beinhaltet alles für Algorithmen
  - Hilfsmittel für Datenstrukturen
  - Ablaufsteuerung
- Unterschiedliche Abstraktionsstufen
- Unterschiedliche Paradigmen

Sprachstufen

- *Maschinensprache*: Direkte Entsprechung des Befehls zu einer Aktion des Gerätes
- *Maschinenunabhängig*: Generalisierte Befehle, direkt in wenige Maschinenbefehle umsetzbar
- *Strukturiert*: Einführung von Variablen, Kontroll- und Datenstrukturen
- *Problemorientiert*: Sprache ganz an die Aufgabe angepaßt, stark spezialisiert

Sprachparadigmen

- *Deklarativ*: Beschreibung des Ziels
- *Imperativ*: Schrittweise Wegebeschreibung
- *Funktional*: Mathematische Ersetzungen
- *Logisch*: Erstellung von Regelwerken
- *Objektorientiert*: Kommunikation von Objekten
- *Prozedural*: Modularisierung, Funktionen
- *Generisch*: Vorlagen für Programmbausteine

Typsystem

- Namen für Arten von Strukturen im RAM
- Erlauben/verhindern Operationen mit Variablen
- *statisch/dynamisch*: Wann sind Typen bekannt?
- *stark/schwach*: Typumwandlung automatisch?
- *sicher/unsicher*: Fehlberechnung möglich?
- Verschiedene Arten: primitiv, zusammengesetzt, Funktionen, polymorph, Transformatoren
- Haskell: statisch, stark, sicher / C: statisch, schwach, unsicher

Beispiele von Typen

- Primitive Typen: maschinennah, unzerlegbar
  - Char, Int, Integer, Float, Double
- Zusammengesetzte Typen: Speicherbereiche
  - Pointer, Struct, Union, Array, List, Tree
- Funktionen: Wandelt Werte in andere Werte um
  - ord: Char -> Int, plus: Int -> Int -> Int
- Polymorph: gilt für viele Typen / show: a -> String
- Transformator: Wandelt Typen in Typen um

Syntax und Semantik

- *Syntax* liegt als kontextfreie Grammatik vor, damit Definition der zulässigen Programme
- *Semantik* Es existiert für jede Grammatikregel eine Beschreibung der Wirkungsweise, damit Definition der Bedeutung der Programme
- http://www.cs.man.ac.uk/~pjj/bnf/c_syntax.bnf
- http://haskell.org/onlinereport/syntax-iso.html

Standardbibliothek

- Normierte Schnittstellen zum Umwelt, insbesondere Ein-/Ausgabe
- Einheitlicher Zugriff auf maschinenabhängige Konstanten und Methoden
- Häufig benötigte Datenstrukturen
- Häufig benötigte Algorithmen
- Standardbibliothek kaum von der Sprache zu trennen

Fehlersuche

- Kein Programm ist fehlerfrei
- Fehler systematisch eingrenzen!
- Testfälle schaffen, Tests oft wiederholen
- Annahmen über bestimmte Zustände explizit machen, z.B. mit assert()
- Entscheidende Stellen im Algorithmus mit Ausgaben versehen
- Fehlerfreiheit beweisen

Etwas anders programmieren

- Lösungen für ein Problem zum Vergleich
  - http://www.99-bottles-of-beer.net/
- Esoterische Programmiersprachen
  - http://esoteric.voxelperfect.net/wiki/Main_Page
  - Empfehlenswert: *Brainfuck, Piet, Whitespace*