Programmiersprachen

Vorlesung 1 vom 16.10.23

Einführung

Christoph Lüth

Universität Bremen

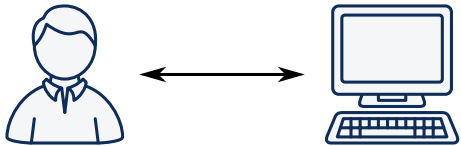
Wintersemester 2023/24

Einführung

Worum geht es?

- Es gibt über 700 Programmiersprachen (sagt Google)
- Wie kriegen wir da Ordnung rein?
- Zugrundeliegende Prinzipien
  - Was haben alle Programmiersprachen gemein?
  - Wo gibt es Unterschiede?
  - **Taxonomie** der Programmiersprachen
- Neue Programmiersprachen lernen (neue, alte, merkwürdige)

Warum Programmiersprachen?



- Wollen Programme in **verständlicher** Notation aufschreiben
- Maschine soll sich dem Menschen anpassen (nicht umgekehrt)
- Programme müssen **maschinenlesbar** und **auführbar** bleiben
- **Modellbildung** und **Abstraktion**

Konzept

Konzept der Veranstaltung

- Integrierter **Kurs** mit zwei Terminen pro Woche:
  - Montag 10- 12 MZH 1100
  - Donnerstag 08:30- 10 MZH 1100
- **Übungen**:
  - Ein Übungsblatt pro Woche.
  - Wird teilweise in den Terminen behandelt, Abgabe bis nächste Woche Montag.
  - **"Leichtgewichtige"** Korrektur.
  - Abgabe und Korrektur über FB3-gitlab.

  Bitte `clueth@uni-bremen.de` als Developer in euer Repo einladen!
- **Referate**:
  - Ab 11. Woche (*i.e.* ab 2024)
  - Studierende stellen je **eine** neue Sprache vor

Scheinbedingungen

1. Mind. 75% der **Übungsblätter** korrekt bearbeitet (50% oder mehr).
2. **Referat** über eine neue Sprache.
3. Mündliche **Prüfung** am Ende.

- Note: 25% Referat, 75% Prüfung

Welche Sprachen betrachten wir?

- Laufende Beispiele:
  - C
  - Rust
  - Java
  - Python
  - Haskell
- Weitere in den Referaten

Liste weiterer Sprachen

- Logische Programmierung: Prolog, Oz
- Dynamisch: JavaScript
- Nebenläufig/Reaktiv: Erlang, Golang
- Abhängige Typen: Idris, Agda, Liquid X (Dependent types)
- Prozedural: Julia, Kotlin, Swift
- Skriptsprachen: Lua, Tcl, sh/bash
- Funktional: SML, OCAML, Elm, Clojure, LISP, Scala
- Stack-basiert: Forth
- Historisch: COBOL, Algol-68, APL, Ada, Smalltalk
- Datenflusssprachen: Id, Lucid, Lustre
- DSLs: R, SQL, Postscript, TeX, Verilog/VHDL, SystemC, SpinalHDL

Struktur der Veranstaltung

- Einführung
- Einfache Ausdrücke und Werte
- Einfache Typen, Anweisungen und Seiteneffekte
- Variablen und Speichermodelle
- Aggregierende Typen
- Ausnahmen und Fehlerbehandlung
- Prozeduren und Funktionen
- Fortgeschrittene Typsysteme
- Datenabstraktion
- Programmierparadigmen
- Skriptsprachen
- Ab Woche 11 (2024): Studentische Vorträge.

Literatur und Basis

- David A. Watt: **Programming Language Design Concepts**, John Wiley & Sons, 2004.
- Maurizio Gabbrielli, Simone Martini: **Programming Languages: Principles and Paradigms**. Springer, 2010.
- Robert W. Sebesta: **Concepts of Programming Languages**. Pearson Education, 2016.
- Norman Ramsey: **Programming Languages. Build, Prove and Compare**. Cambridge University Press, 2023.

Vorkenntnisse

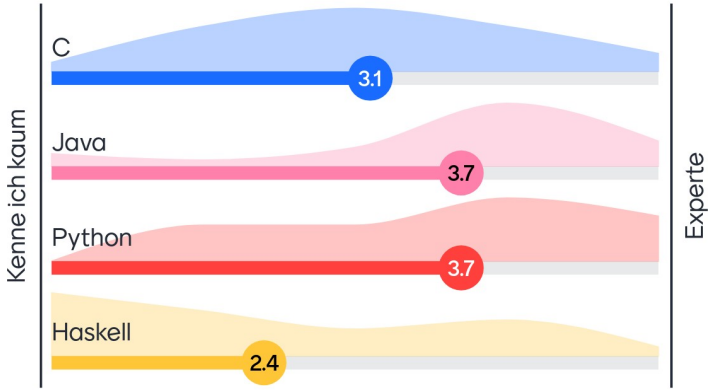
Online-Umfrage:
`https://www.menti.com/al8j27pcua32`

- Vorkenntnise in folgende Sprachen:
  - C
  - Java
  - Python
  - Haskell
- Welche weiteren Sprachen kennt ihr?

Ergebnisse I

Mentimeter

Vorkenntnisse



Ergebnisse II

Mentimeter

Welche anderen Sprachen kennt ihr?

66 responses

Grundlagen & Geschichte

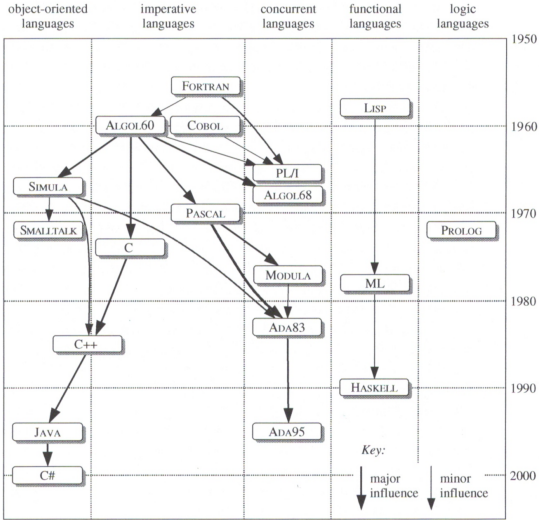
Was ist eine Programmiersprache?

1. Definierte, maschinenlesbare **Syntax**
2. Mathematisch, informell oder pragmatisch definierte **Semantik**
3. Die Sprache muss **ausführbar** und **Turing-mächtig** sein

Arten von Programmiersprachen

- Programmiersprachen sind immer **Abstraktionen** über einem Berechnungsmodell.
- **Imperativ**: Zustandsübergänge auf einem Speicher (Turing-Maschine)
  - Abstraktion durch Datentypen
  - Abstraktion durch Verkapselung
- **Funktional**: Rekursive Funktionen (Auswertung von Ausdrücken)
- Sonstige: logische, domänenspezifisch, Quantencomputer, ...

Historisches: Stammbaum einiger Programmiersprachen



Was ist ein Computer?

- Ein Computer:
  1. ist elektronisch und digital;
  2. beherrscht die vier arithmetischen Operationen ($+$, $-$, $\cdot$, $/$);
  3. kann programmiert werden;
  4. erlaubt die Speicherung von Programmen und Daten.
- (2)– (4) garantiert **Turing-Vollständigkeit**.

Die ersten Computer

- Zuse Z3 (Zuse, 1941): sogar mit Fließkommarithmetik, aber nicht elektronisch;
- ENIAC (Mauchly, Eckert, von Neumann, 1946): nicht reprogrammierbar
- EDSAC (Wilkes, 1949): erfüllt alle vier Kriterien
- Thomas Watson, IBM (1943): “Es gibt einen Weltmarkt von 5 Computern”

Generationen von Programmiersprachen:

- Programmiert in **Maschinensprache**
  - Programmiersprachen der ersten Generation — 1 GL
- Symbolische Repräsentation: **Assemblersprachen** (2 GL)
- Hochsprachen: 3GL
- 4GL: Nicht exakt definiert
  - “Programming without the Programmer”, CASE, . . .
  - Model-driven development, DSLs

Die erste Hochsprachen: FORTRAN

- Hardware war **teuer** als Arbeitskraft — deshalb **Programmeffizienz** wichtig
- FORTRAN (1957): FORmula TRANslator
- Erste Programmiersprache mit symbolischer Notation `a*2+ b`
- Entwickelt für numerische Berechnungen
- Lief auf der IBM 704

John Backus
(1924–2007)

- Turing Award (1977)
- FORTRAN
- Backus-Naur-Form

Die ersten Hochsprachen: COBOL

- COBOL (1960): COmmon Business Oriented Language
- Erste Sprache für Business-Anwendungen
- Standardisiert, Design by committee

Grace Hopper
(1906–1992)

- Erfand “Bugs”
- Allgemeinverständliche Programmierung
- Beeinflusste COBOL

Die ersten Hochsprachen: ALGOL

- ALGOL (Algorithmic Language): Ursprung der modernen Programmiersprachen
- Erste Version 1958, 1960 (ALGOL-60), spätere Versionen (ALGOL-68)
- Amerikanisch-Europäische Koproduktion
- Erste Sprachen mit formal definierter Grammatik (BNF), Blöcken, strukturierter Programmierung, call-by-name
- "ALGOL-Syntax":

```
for p := 1 step 1 until n do
  for q := 1 step 1 until m do
    if abs(a[p, q]) > y then
      begin
        y := abs(a[p, q]);
        i := p; k := q
      end
```

Funktionale Sprachen: LISP

- LISP (LISt Processor):
  die erste funktionale Sprache
- 1960 von John McCarthy am MIT entworfen
- Speziell für nicht-numerische Probleme (KI)

- Basiert auf dem Lambda-Kalkül
- Alles ist eine S-Expression

LISP-Maschine:
Symbolics 3640

John McCarthy
(1927–2011)

- Turing Award (1971)
- LISP
- Pionier der KI

Die 1970er Jahre

- C (Dennis Ritchie & Ken Thompson, 1972):
  - Portable Assembler-Sprache
  - Implementationssprache für UNIX
- Pascal (Niklaus Wirth, 1970):
  - Virtuelle Maschine (P-Code)
  - Strukturiert, block-orientiert, stark getypt
- Smalltalk (Alan Kay, 1970):
  - Objektorientiert, GUI integriert
- ML (Robin Milner, 1974):
  - Für den LCF-Beweiser, Hindley-Milner-Polymorphie
  - Standardisiert (1983), mathematisch formal definierte Semantik (1997)
- Prolog (Bob Kowalski, 1974):
  - Logische Programmierung, Ausführung durch **Resolution**

Die 1980er und 1990er Jahre

- C++ (Stroustrup, 1986): objektierientierte Erweiterung von C
- Ada (1983): vom DoD standardisiert, sehr komplexer Standard.
  - Erster Compiler 1986
- Erlang (Armstrong, 1986–92): für verteilte und nebenläufige Applikationen, Fa. Ericsson
- Java (Gosling *et al*, 1990): zuerst "Oak", für Set-Top-Boxen.
  - Objektorientiert, JVM, Applets — portabel und sicher
- Haskell (1987, 1998): nicht-strikt und funktional,
- Python (van Rossum, 1991): leichtgewichtig, einfach zu nutzen
- JavaScript (Eich, 1995): Skriptsprache für den Browser

Wie geht's weiter?

- Einführung
- Einfache Ausdrücke und Werte
- Einfache Typen, Anweisungen und Seiteneffekte
- Variablen und Speichermodelle
- Aggregierende Typen
- Ausnahmen und Fehlerbehandlung
- Prozeduren und Funktionen
- Fortgeschrittene Typsysteme
- Datenabstraktion
- Programmierparadigmen
- Skriptsprachen
- Ab Woche 11 (2024): Studentische Vorträge.