Todos

Algorithmen und Programmiersprachen

Prof. Dr. Christian Becker

Universität Stuttgart, Institut für Parallele und Verteilte Systeme

15. Mai 2025

Motivation

- Wir haben letztes Mal gesehen, wie Maschinensprache aussieht
- Arbeiten in Maschinensprache ist möglich.. aber
  - fehleranfällig
  - zeitintensiv
- Das muss doch besser gehen!

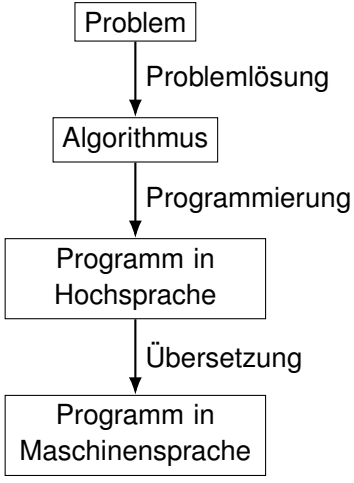
Bild aus Kap 3

Motivation

- Ein Programm ist eine Reihe von Anweisungen
- Maschinensprache ist an eine Hardwarearchitektur gebunden
- Maschinensprache ist für Menschen schwer begreifbar

Gibt es Erkenntnisse zur systematischen, hardwareunabhängigen Programmierung und Softwareentwicklung?

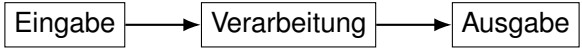
Inhalt dieser Vorlesung



Algorithmus

Definition (Algorithmus)

Ein Algorithmus ist eine wohldefinierte Berechnungsvorschrift, die in endlich vielen Schritten auf Basis von Eingabewerten Ausgabewerte berechnet.



- Ein Algorithmus ist eine abstrakte Beschreibung einer Problemlösung
- Beschreibt eine systematische und schrittweise Vorgehensweise

Probleme und Algorithmen

- Algorithmen lösen nicht eine einzelne Probleminstanz, sondern eine ganze Klasse von Problemen
- Die Instanz ist durch Parameter [paˈʁaːmetɐ] spezifiziert

Beispiele

- Berechne Fakultät von n (Problem),
  Berechne 4! (Instanz)
- Berechne den kürzesten Weg von A nach B (Problem),
  Berechne den Weg von Stuttgart nach Frankfurt (Instanz)

Algorithmen sind ein fundamentales Konzept

Ein Algorithmus ist unabhängig von Programmiersprachen und Hardware

- Ein Algorithmus kann für Computer *implementiert* werden
- Nicht nur Computer können Algorithmen ausführen, auch Menschen lösen Probleme mit Algorithmen (z.B. schriftlich multiplizieren, addieren, ..)

Algorithmen sind kreativ: Es gibt viele Algorithmen für das gleiche Problem!

Algorithmen

- Wir betrachten Probleme, für die es verallgemeinerbare Lösungsansätze gibt

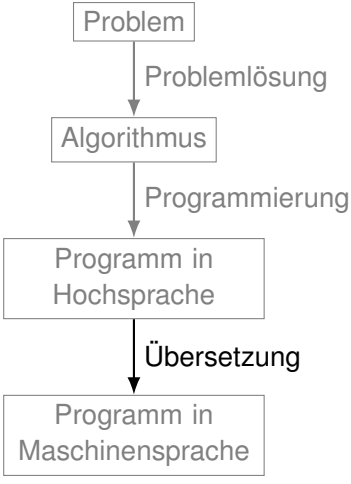
Beispiele für Algorithmen und deren Schritte

-

Hochsprachen

- Idee: Daten und Rechenanweisungen abstrakt darstellen
- Nicht mehr mit Details wie Registern kämpfen
- Werkzeuge benutzen die abstrakte Beschreibung in Maschinensprache übersetzen

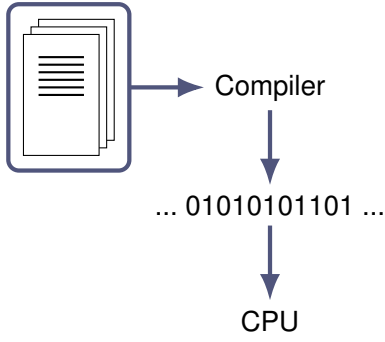
Übersetzen in Maschinensprache



- Wir schauen an, wie Übersetzer aussehen
- Unterschiede zwischen Übersetzern
- Vor- und Nachteile, ...

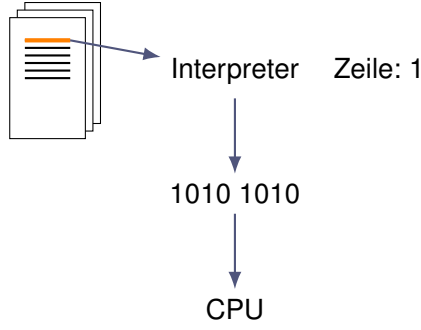
Compiler

- Das Programm wird **komplett** in Maschinensprache übersetzt
- Programm kann nach und während der Übersetzung stark optimiert werden



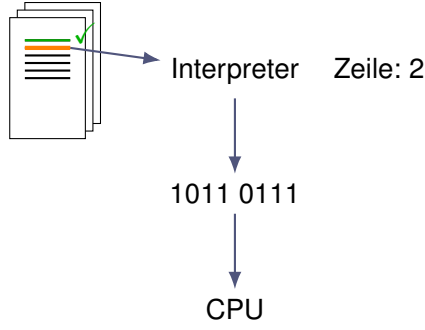
Interpreter

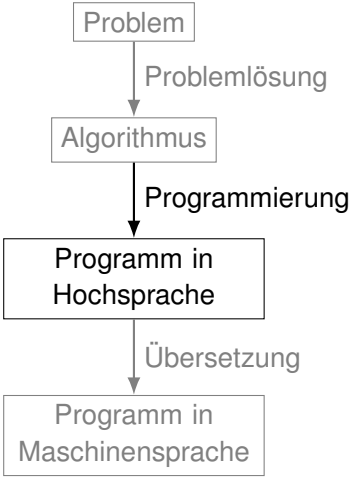
- Das Programm wird **während der Ausführung stückweise** in Maschinensprache übersetzt
- Analog zu einem Simultanübersetzer der satzweise übersetzt
- Jede Anweisung aus dem Programm wird bei Ausführung (möglicherweise neu) übersetzt
- Mögliche Optimierung: Merken bereits übersetzter Anweisungen



Interpreter

- Das Programm wird **während der Ausführung stückweise** in Maschinensprache übersetzt
- Analog zu einem Simultanübersetzer der satzweise übersetzt
- Jede Anweisung aus dem Programm wird bei Ausführung (möglicherweise neu) übersetzt
- Mögliche Optimierung: Merken bereits übersetzter Anweisungen

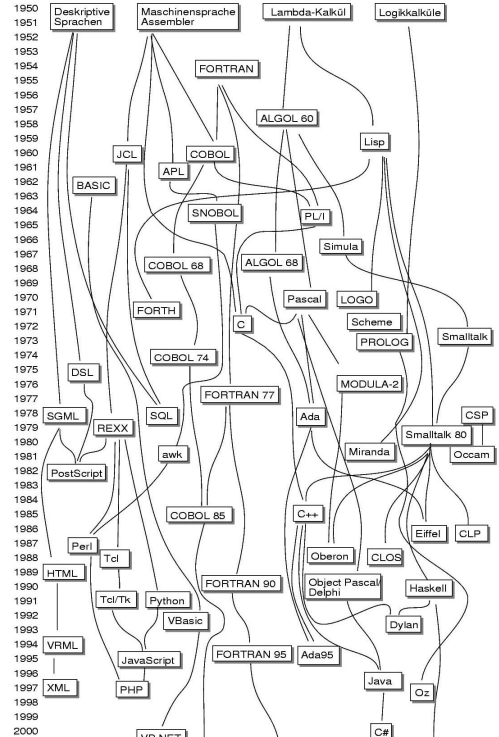




- Welche Möglichkeiten für Hochsprachen (das was übersetzt wird) gibt es?
- Gibt es eine Sprache für jedes Problem?
- Wann wähle ich welche Sprache?

Was übersetze ich denn nun?

- Das hängt vom Problem ab.
- Es gibt viele Programmiersprachen
- Viele ähneln sich, manche sind speziell für bestimmt Anwedungsfälle entworfen..
- Hier konzentrieren wir uns auf einen Teil des Sprachbaums


1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
Deskriptive Sprachen
Maschinensprache Assembler
Lambda-Kalkül
Logikkalküle
FORTRAN
ALGOL 60
Lisp
JCL
COBOL
APL
BASIC
SNOBOL
PL/I
Simula
COBOL 68
ALGOL 68
FORTH
Pascal
LOGO
C
Scheme
PROLOG
Smalltalk
COBOL 74
DSL
FORTRAN 77
MODULA-2
SGML
REXX
SQL
Ada
CSP
Smalltalk 80
Miranda
Occam
awk
PostScript
COBOL 85
C++
Eiffel
CLP
Perl
Tcl
Oberon
CLOS
HTML
FORTRAN 90
Object Pascal Delphi
Haskell
Tcl/Tk
Python
VBasic
Dylan
VRML
JavaScript
FORTRAN 95
Ada95
Java
Oz
XML
PHP
C#

Programmiersprachen: Ein bisschen Geschichte

Fortran: 1950er Jahre

```fortran
S = (IA + IB + IC) / 2.0
AREA = SQRT ( S * ( S - IA ) * ( S - IB ) * ( S - IC) )
WRITE ( 6, 601 ) IA, IB, IC, AREA
```

Fortran

- FORmular TRANslator
- Wissenschaftliches Rechnen
- Erweiterung durch moderne Konzepte mit den Jahren

Programmiersprachen: Ein bisschen Geschichte

LISP: 1950er Jahre

```lisp
(+ 2 3 4) ;; Addiere 2, 3 und 5
(setf p 4.1415) ;; Setze die Variable p auf 3.1415

(defun square (x) ;; Defniere eine Funktion die ihr Argument quadriert
  (* x x))

;; Quardiere die Zahl 4
(square 4)
```

Lisp

- LISt Processing
- Formale Grundlage im λ-Kalkül
- Grundlage erster Systeme in der KI, Theorem-Beweiser, Expertensysteme

Programmiersprachen: Ein bisschen Geschichte

COBOL: 1950er Jahre

```cobol
.
        identification division.
        program-id. display-test.

        procedure division.

        main-procedure.
            display "hello world"
                line 07 column 05
                with blank line
            end-display
            goback.

        end program display-test.
```

Cobol

- COmmon Business Oriented Language
- Unterstützung betriebswirtschaftlicher Anwedungen
- Modellierung von Daten

Programmiersprachen: Ein bisschen Geschichte

C: 1970er Jahre

```c
int main() {
    printf("Hello World!\n");
}
```

C

- Ziel: Systemsoftware (Betriebssysteme) portabel erstellen
- Leichtgewichtig und effizient übersetzbar
- Objektorientierte Erweiterung (C++: 1980er Jahre)

Programmiersprachen: Ein bisschen Geschichte

Java: 1990er Jahre

```java
public static void main(String[] args) {
    System.out.println("Hello World!");
}
```

Java

- Zentraler Fokus unserer Vorlesung

Programmiersprachen: Ein bisschen Geschichte

Python: 1990er Jahre

```python
def main():
    printf("Hello World!\n");

if __name__ == "__main__":
    main()
```

Python

- Nicht Fokus der Vorlesung, sollten Sie aber unbedingt lernen!

Programmiersprachen: Ein bisschen Geschichte

Go: 2000er Jahre

```go
import "fmt"

func main() {
    fmt.Println("hello world")
}
```

Go

- Moderne Multitasking-Ansätze als Kern der Sprache

Programmiersprachen: Ein bisschen Geschichte

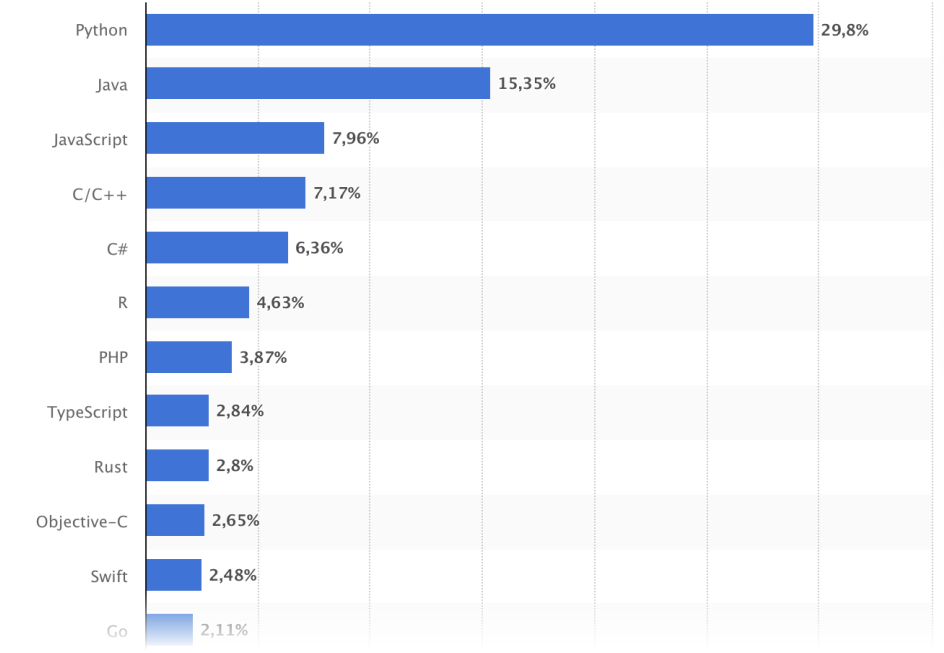
Rust: 2010er Jahre

```rust
fn main() {
    println!("Hello World!");
}
```

Rust

- Speichersicherheit als Kern der Sprache

Beliebte Programmiersprachen



Java

- Sprache aus der Familie C, C++
- Java Virtual Machine: Versteckt zugrundeliegende Hardware, Programm läuft einfach so auf jedem Rechner auf dem die JVM läuft
- Typisiert, Objektorientiert, Garbage collected, ... (dazu später mehr)
- Gut um Programmieren zu lernen, auch sehr relevant in vielen Anwendungsbereichen. Cloudanwendungen sind gerne in Java geschrieben, einige Android-Apps sind in Java, ...
- Am Ende: Wer Java kann, lernt schnell auch andere Sprachen

Unsere Mission

- Programmieren lernen
- Denken lernen wie eine Informatikerin
- Wir schauen uns an
  - Klassische Probleme wie Suchen, Sortieren, ..
  - Dazu geeignete Datenstrukturen
- Das ist ein Anfang auf einem langen Weg der Programmierung und Softwareentwicklung
- Programmieren ist ein Handwerk: Üben übt!