

# Einführung

## Einführung in die Programmierung

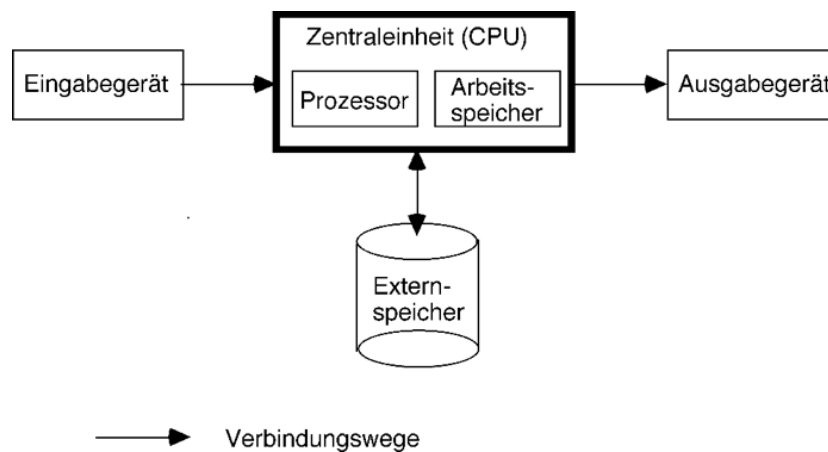
Johannes Brauer

31. Dezember 2019

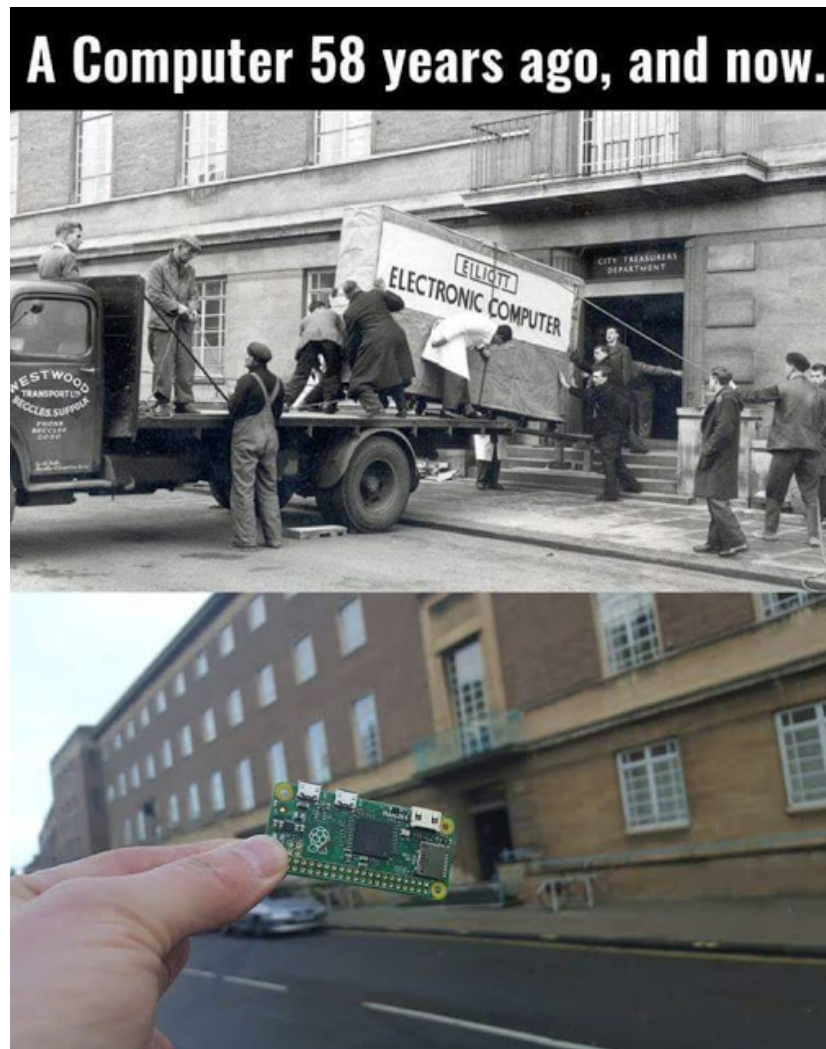
### Programme für Computer

Ein Programm ist eine Folge von Anweisungen (Befehlen) an eine Maschine (Rechner, Computer), die von dieser „verstanden“ wird und damit ausgeführt werden kann.

### Aufbau von Computern



## Computer im Wandel der Jahrzehnte



- Ein iPhone enthält ca. 1 Milliarde Transistoren.
- Um diese Rechenleistung mit der Technologie der 1950er Jahre zu bauen, bräuchte es:
  - 1 Milliarde Elektronenröhren
  - 170 vehicle assembly buildings, um sie unterzubringen
  - 1 Terawatt Leistung, um sie zu betreiben
  - das entspräche 500 2-Gigawatt-Kernkraftwerken für ca. 50 Milliarden Euro
  - das entspräche dem Weltbruttosozialprodukt von 60 Jahren
- Smartphones realisieren eine Steigerung der Rechenleistung um den Faktor  $10^{22}$  verglichen mit der Technologie vor 60 Jahren.

Welche Fortschritte gibt es in dieser Zeit in der Software?

## Was können Computer?

- **Problem:** Computer können nur sehr simple Dinge tun.
- Beispiel: Der Computer soll 10 mal „piepen“. Pseudo-Maschinenprogramm

```
        put the number 10 into memory location 0
a   if contents of location 0 is negative go to line b
        beep
        subtract 1 from the number in location 0
        go to line a
b   ... rest of program ...
```

- Man stelle sich vor, auf diese Weise ein Programm für die Tourenplanung einer Spedition zu schreiben.
- Fällt Ihnen an dem Programm etwas auf?
- Besser wäre, man könnte z. B. schreiben:

```
(dotimes [n 10] (beep))
```

## Programmiersprachen

### Was ist eine Programmiersprache?

- Damit die Maschine uns „versteht“, müssen Programme in einer für sie verständlichen Sprache formuliert werden.
- Programmiersprachen sind *formale Sprachen* zur Formulierung von Programmen, die auf Rechnern ausführbar sind.

### Maschinenorientierte Programmiersprachen

- Zu jeder Maschine gehört eine Liste von Dingen, die sie tun kann:
  - Wasserkocher?
  - MP3-Player?
  - Computer?
- Die vollständige Liste der Dinge (Befehle), die ein Computer tun kann, kann als seine *Maschinensprache* (machine language) bezeichnet werden.
- Maschinencode: interne (ausführbare) Darstellung eines Maschinenprogramms als Bitmuster.
- Assemblersprache (assembly language): Symbolische, textorientierte Darstellung einer Maschinensprache. Ihre Merkmale sind:
  - Die Liste der Befehle ist dieselbe, wie die der Maschinensprache.
  - Symbolische Namen der Befehle
  - Dezimalzahlen, symbolische Adressen.

- Assemblerprogramm, Assemblercode: Programm in Assemblersprache.

BINARY		SOAP	
1999015201	5801610167	STL	-1
0170017301	7601820185	RAU	8005
0141014401	5001530156	MPY	*
0154020402	0501630254	ALO	-1
0211014201	4901720143	SLT	0004
0203020602	1901770235	ALO	8002
0192019902	2201930253	RAU	0115
0256015101	5902610224	STU	11
0137021301	6902270183	RAL	8006
0201030401	5702630166	STL	-1
0272017501	8101840187	RAU	8007
0160021703	2202430146	MPY	*
0162036902	3102330136	ALO	-1
0155016402	6002520202	SLT	0004
0197018802	2802290319	ALO	8002
0000000000	0000001830	RAU	0096
6501640167	6901701822	STU	12
6901851822	6501880141	RAU	12
6901561823	8000010162	FMP	11
8800010169	6580070211	FAD	SUM
1500390143	3500040203	STU	SUM
6580060235	2000390192	AXC	0001 \$5
3500040253	1502568002	RSL	8007
3901740224	3202100137	STU	K
2401800183	1501470201	ALO	L
6901661822	6503190272	BMI	+4
6901871823	5200010160	PUNCH	1, SUM, 1, J
4601460204	5000010162		PNCHE
4601360154	0100008000	LDD	
0000000004	0000000005		
0000000179	0000000269		
0000000179	0000001999		

## Problemorientierte Programmiersprachen

- Höhere, problemorientierte Programmiersprache: Formale Sprache zur textuellen Darstellung von Programmen, deren Konstrukte
  - mächtiger als einzelne Maschinenbefehle sind (kürzere „Befehlsliste“),
  - Details der von-Neumann-Architektur verbergen,
  - die Formulierung von Algorithmen unabhängig von einem bestimmten Rechensystem ermöglichen,
  - sich an den Bedürfnissen eines Anwendungsbereichs orientieren.
- Quellprogramm (source code): Programm in Hochsprache.

## SOAP

```

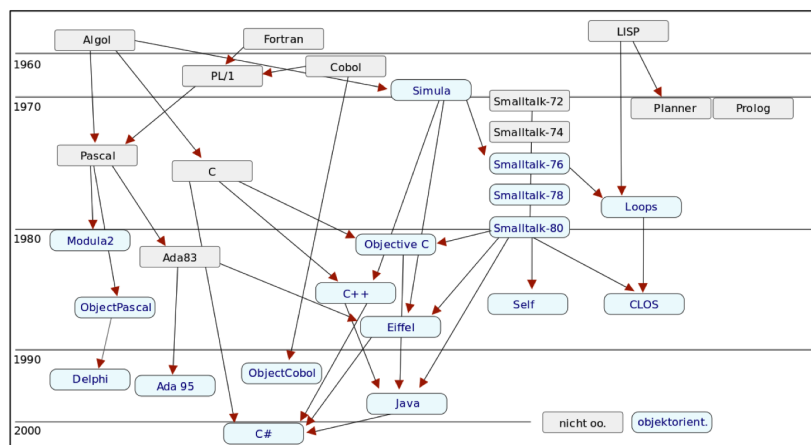
STL      -1
RAU      8005
MPY      +
ALO      -1
SLT      0004      8002
ALO      0115
RAU      11
STU      11
RAL      8006
STL      -1
RAU      8007
MPY      +
ALO      -1
SLT      0004      8002
ALO      0096
STU      12
RAU      12
FMP      11
FAD      SUM
STU      SUM
AXC      0001      $5
RSL      8007
STU      K
ALO      L.
BMI      +4
          PUNCH 1, SUM, 1, J
LDD
          PUNCHF
          J
    
```

## FORTRAN

```

C 0000 RECTANGULAR MATRIX
C 0000 MULTIPLICATION
      DIMENSION A(4,5), B(5,3)
      READ 1, A, B
      READ 1, N, M, L
      DO 4 J= 1, N
      DO 4 I= 1, M
      SUM = 0.0
      DO 3 K= 1, L
      SUM=SUM+A(I,K)*B(K,J)
      PUNCH 1, SUM, I, J
      4 END
    
```

## Die Entwicklung von Programmiersprachen



## Wichtige Programmiersprachen

**1954 - 57 Fortran** (Formula Translation) von J. W. Backus, IBM.

**1956 - 62 Lisp** (List Processing Language) von J. McCarthy. Funktional, Hauptsprache der Künstlichen Intelligenz

**1958 - 60 Algol 60** (Algorithmic Language) von P. Naur u.a.

**1959 - 61 Cobol** (Common Business Oriented Language), noch heute weit verbreitete Sprache für kommerzielle Anwendungen.

**1967 Simula 67** von Dahl/Nygaard, erste objektorientierte Sprache

**1968 - 71 Pascal** von N. Wirth, einfach, strukturierte Programmierung, strenges Typkonzept

**1970 - 72 C** von D. Ritchie, maschinennah, mit Unixverbunden, für Betriebssystemprogrammierung.

**1970 - 80 Smalltalk** von Kay/Goldberg/Ingalls, rein objektorientiert.

**1975 - 80 Ada** von J. Ichbiah/DoD, modular, Prozesse, Ausnahmebehandlung, komplex, militärische Anwendungen.

**1975 - 82 Prolog** (Programming in Logic) von Colmerauer/Warren, modelliert logisches Schließen, KI-Sprache.

**1980 Modula-2** von N. Wirth, modular, für Systemprogrammierung.

**1980 - 86 C++** von B. Stroustrup, objektorientierte Erweiterung von C.

**1985 - 86 Oberon** von N. Wirth, objektorientiert, für Systemprogrammierung.

**1985 - 88 Eiffel** von B. Meyer, objektorientiert

**1996 - Java** objektorientiert, ursprünglich eingetragenes Warenzeichen der Firma Sun Microsystems, heute im Besitz von Oracle

**neuere C#** ähnlich Java. Microsoft, .Net-Plattform

**F#** funktional. Microsoft, .Net-Plattform

**Scala** funktionale Erweiterung von Java

**Dart** Googles JavaScript-Alternative

**Clojure** Lisp-Dialekt auf der JVM

**u.v.a.m.**

## **Syntax, Semantik, Pragmatik von Programmiersprachen**

- Syntax legt fest,
  - welche Sprachelemente und -konstrukte es gibt und
  - wie mit ihrer Hilfe korrekte Sätze in der Sprache formuliert werden
  - Syntax = Menge von Regeln, die die Struktur von Programmen bestimmen.
- Semantik einer Programmiersprache
  - legt die Bedeutung syntaktisch korrekter Sätze fest
  - legt fest, welche Wirkung jedes Sprachelement oder -konstrukt im Programmablauf hervorruft.
  - Semantik = Menge von Verhaltensregeln, die die Funktionsweise von Programmen bestimmen.
- Pragmatik
  - Intention des Programmierers mit einem Programm
  - Nutzen der Ausdrucksmöglichkeiten einer Programmiersprache für die Formulierung von Lösungen

## Zusammenfassung

- **Programmiersprachen** sind formale Sprachen, in denen sich für den Menschen verständliche Programme für Rechenmaschinen formulieren lassen. Wichtig sind ihre Syntax, Semantik und Pragmatik.