

Programmiersprachen  
Vorlesung 10 vom 20.12.21  
Weihnachtsvorlesung

Christoph Lüth

Universität Bremen

Wintersemester 2021/22

### Was machen wir heute?

- ▶ Geschichtliches
- ▶ Eine besondere Sprache

## Geschichtliches

### Was ist ein Computer?

- ▶ Ein Computer:
  - 1 ist elektronisch und digital;
  - 2 beherrscht die vier arithmetischen Operationen (+, -, ·, /);
  - 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

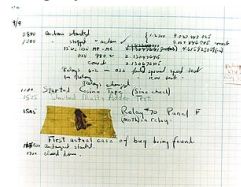


John Backus  
(1924–2007)

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

### Die ersten Hochsprachen: COBOL

- ▶ COBOL
- ▶ 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
    end
  end
end Absmax
```



## 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): objektorientierte 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



## Eine besondere Sprache



## Brainfuck

- ▶ Brainfuck wurde 1993 von Urban Müller erfunden, um den "kleinstmöglichen Compiler für eine Turing-vollständige Sprache" zu schreiben.
- ▶ Abgeleitet von  $P'$  (Corrado Boehm, 1964)
- ▶ Acht Kommandos, Turing-vollständig
- ▶ Kryptisch und von keinem praktischen Nutzen



## Ein Beispielprogramm

- ▶ Hello, world:

```
+++++++
[>++++[>+>++++>++++>+<<<<-] >+>+>+>+ [<] <-]
>> .>---.+++++++ .+++ .>> .<-.
< .+++ .----- .----- .>> .>+.
```



## Die Sprache

- ▶ Syntax: acht Kommandozeichen
- ▶ Lexikalik: alles andere ist Kommentar
- ▶ Ausführungsmodell:
  - ▶ Maschine mit Programm und Instruktionszeiger, Datenzeiger und 30.000 Speicherzellen



## Kommandos

<	Datenzeiger erhöhen
>	Datenzeiger erniedrigen
+	Aktueller Zellenwert erhöhen
-	Aktueller Zellenwert erniedrigen
.	Aktueller Zellenwert ausgeben
,	Aktueller Zellenwert einlesen
[	Springt hinter das entsprechende ] wenn aktueller Zellenwert 0 ist
]	Springt hinter das entsprechende [ wenn aktueller Zellenwert nicht 0 ist

▶ [P] ist Iteration von P solange aktueller Zellenwert ungleich 0 ist.

## Einfache Programme:

- ▶ `, [ . , ]` Echo
- ▶ `[>+<-]` Addition  $p[i+1] = p[i+1] + p[i]$
- ▶ `[>-<-]` Subtraktion  $p[i+1] = p[i+1] - p[i]$
- ▶ `>[-]<[>+<-]` Verschieben  $p[i+1] = p[i]; p[i] = 0$

## Größere Programme

### ▶ Kopieren:

```
>[-]>[-]<<      Initialisierung
[>+>+<<-]      Verschiebe p[i] nach p[i+1], p[i+2]
>>[<<+>>-]     Verschiebe p[i+2] nach p[i]
<<
```

## Zusammenfassung

- ▶ Brainfuck ist:
  - ▶ Turing-vollständig
  - ▶ extrem kompliziert zu benutzen
  - ▶ extrem einfach zu implementieren
  - ▶ in der Praxis unbrauchbar

