Programmieren 1

Einführung in die Programmierung, Programmiersprache C

Dozent: Jonas Fritzsch, M.Sc.

Kontakt: fritzsch@lehre.dhbw-stuttgart.de

Organisatorisches

- Vorlesungszeiten, Termine
- Übungen, Projekt
- Webseite zur Vorlesung
  http://wwwlehre.dhbw-stuttgart.de/~fritzsch/Programmieren/
- Skript
- Klausur

Inhalte der Vorlesung

- Grundlegende Konzepte der Programmierung
- Algorithmisches Denken schulen
  Problem → Algorithmus → Programm
- Programmiersprache C
- Grundlegende Algorithmen
  (Felder, Listen, Suchen, Sortieren …)
- Übungen

Literatur

Programmieren in C. ANSI C (2. Ausgabe) / Brian W. Kernighan, Dennis M. Ritchie / Hanser Fachbuch

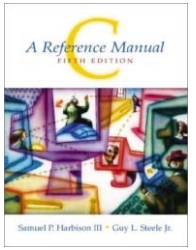
C von A bis Z / Wolf / Galileo Computing
http://openbook.galileocomputing.de/c_von_a_bis_z/

C als erste Programmiersprache / Manfred Dausmann et al. / Vieweg+Teubner Verlag

C Programmieren von Anfang an / Helmut Erlenkötter / rororo Computer

Practical C Programming / Steve Qualline / O'Reilly

C: A Reference Manual, Fifth Edition / Harbison, Steele / Prentice Hall



Teil I: Einführung

Gliederung

Allgemeines

Maschinen und Sprachen

Algorithmen

Allgemeines

Was ist Informatik? (Information – Automatik, Mathematik)

Def. 1 Informatik ist die Wissenschaft von der **systematischen Verarbeitung von Informationen** - insbesondere deren maschinelle (automatische) Bearbeitung, Speicherung und Übertragung durch Digitalrechner (Computer). *(Informatik Duden)*

Def. 2 Informatik umfasst die **automatisierte Informationsverarbeitung** in Natur, Technik und Gesellschaft *(Fachliteratur "Grundlagen d. Informatik")*

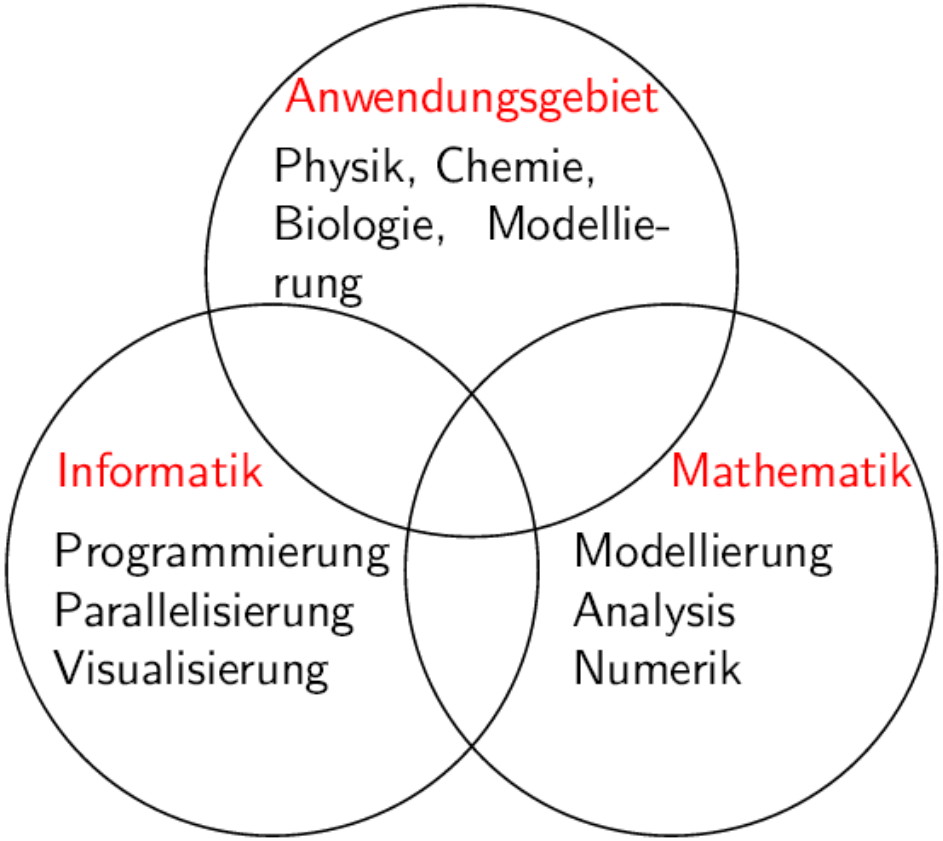
- Begriff 1957 von Karl Steinbuch eingeführt
- vgl. "computer science"

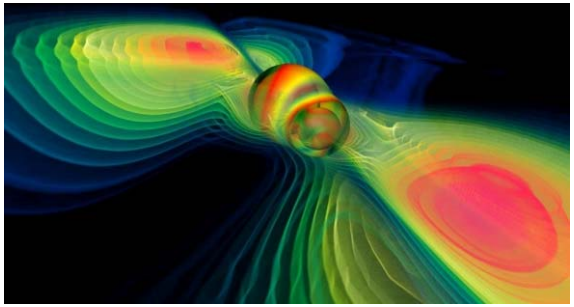
  E. W. Dijkstra: *„Computer Science is no more about computers than astronomy is about telescopes“*

  → Informatik vs. Rechnertechnik
- Ende der 1960er Jahre erstmals Studienrichtung

Interdisziplinäre Forschungsdisziplin

Beispiel: Wissenschaftliches Rechnen in Simulationen
Ziel: Vermeidung teuer Experimente

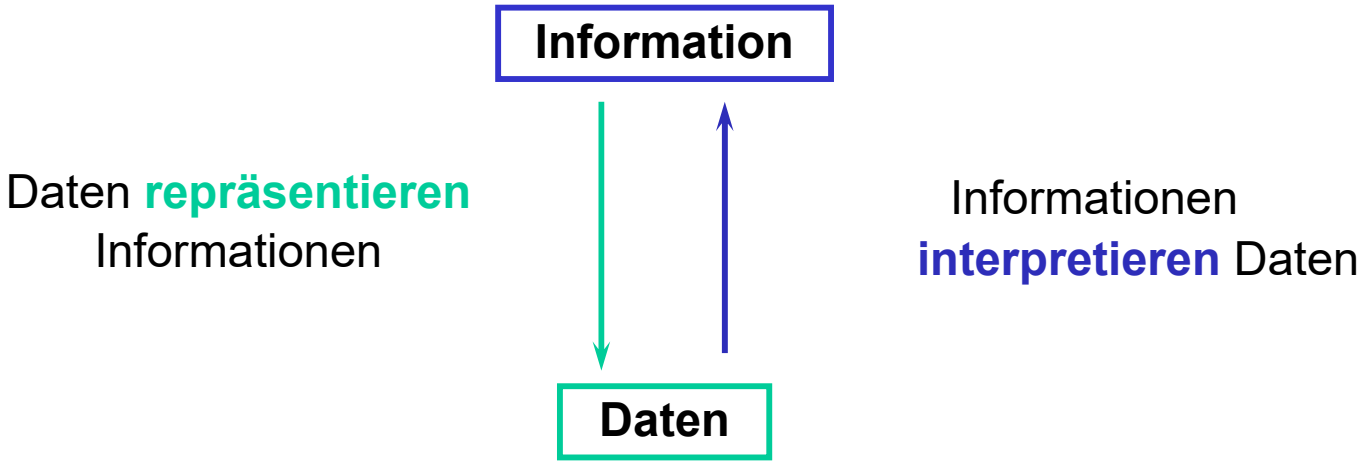




Teilgebiete der Informatik

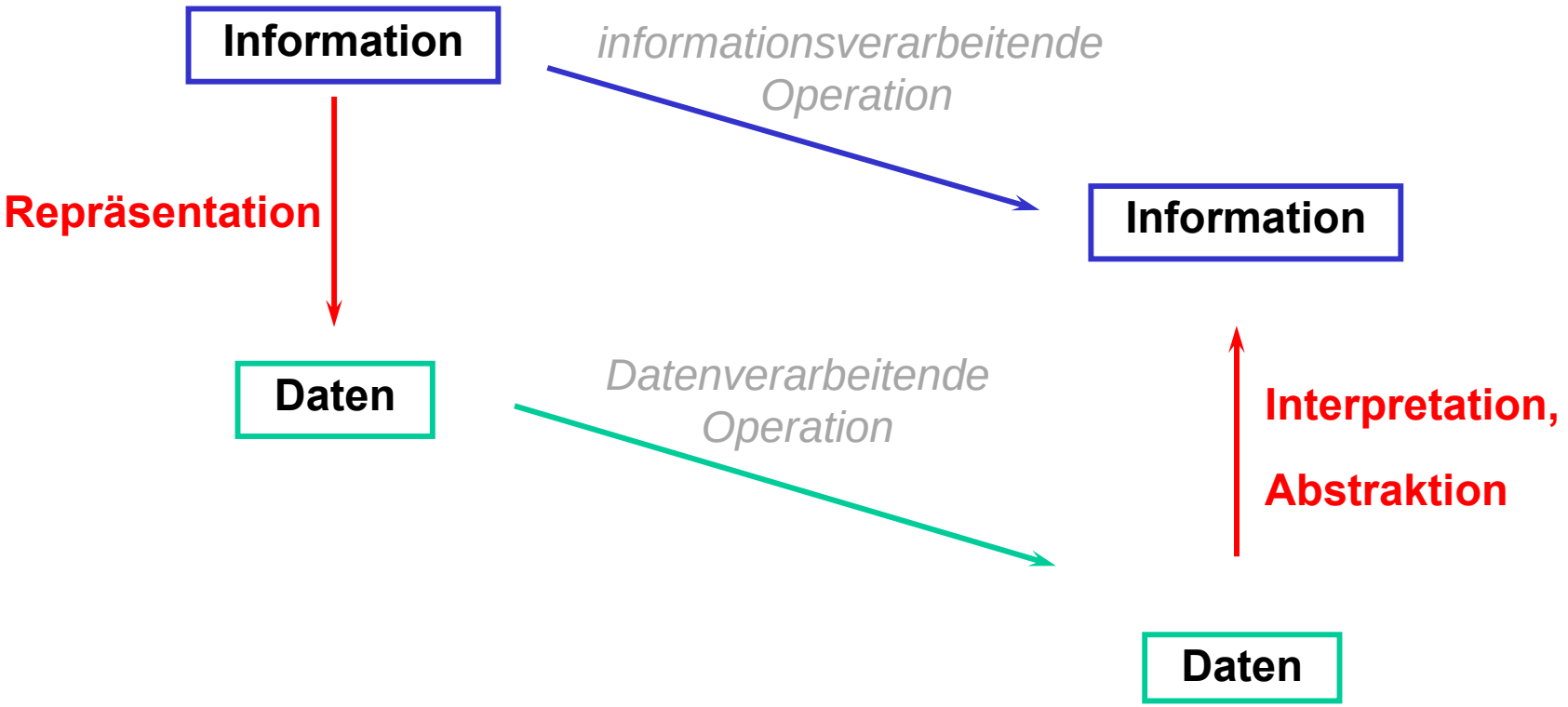
Theoretische Informatik	Praktische Informatik	Angewandte Informatik	Technische Informatik
Sprachen u. Automaten	Programmier-sprachen	Ingenieur-bereich	Rechnertechnik
formale Logik	Compiler Interpreter	Wirtschaft / Verwaltung	Rechner-architektur
Berechen-barkeit	Info- / DB-Systeme	Medizin / Med.Technik	Prozeßdaten-verarbeitung
Komplexitäts-Theorie	Betriebs-systeme	Methoden Entwicklung	Mikroelektronik
Informations-/ Codierungsth.	Künstl. Intelligenz		

Information und Daten



Information ≙ Bedeutungsgehalt von Zeichen, Nachrichten

Informationsverarbeitung und Datenverarbeitung



Bits

Informationen werden repräsentiert als Folgen von Bits. **Bit = B**inary Dig**it**

Verwendung:
- Binärziffer
- Maßeinheit für Datenmenge
- Maßeinheit für Informationsgehalt

- Wert 0 oder 1 / aus oder an / ja oder nein
- elektrische Spannung: 0 = 0 Volt 1 = 5 Volt
- Magnetisierung: 0 = entmagnetisiert 1 = magnetisiert

Bitfolgen

- mehr als 2 Zustände abbilden
- Bitfolge aus 2 Bits = 4 Zustände

  00 01 10 11

  → es gibt genau 2^N mögliche Bitfolgen der Länge N Bit

Beispiel: Windrichtung

Frage: Aus welcher Himmelsrichtung weht der Wind?

Codierung von Windrichtungen:

Nord
Süd
Ost
West
Nordwest
Nordost
Südwest
Südost

Hexziffern

- Folgen von Nullen/Einsen sind unübersichtlich

```
01001111011000010110110001101100
```

- Idee: Anordnung in Gruppen zu 4 Bits

```
0100 1111 0110 0001 0110 1100 0110 1100
```

- **Halb-Byte:** 16 Zustände
- Ziffern '0' bis '9' und Buchstaben 'A' bis 'F'

0000 = 0	0100 = 4	1000 = 8	1100 = C
0001 = 1	0101 = 5	1001 = 9	1101 = D
0010 = 2	0110 = 6	1010 = A	1110 = E
0011 = 3	0111 = 7	1011 = B	1111 = F

Bytes

- Oktett von Bits: **8 Bits = 1 Byte**

- Beispiel: $(11000101)_2 = (C5)_{16} = (197)_{10}$

Beispiele der Codierung:
- Zahl zwischen 0 und 255
- Zahl zwischen -128 und +127
- Zeichen im Zeichencode, z.B. ASCII Code

Einheiten:

1 Kilobyte (neu kibibyte)	2^{10}	1024 Bytes
1 Megabyte (neu mebibyte)	2^{20}	1.048.576 Bytes
1 Gigabyte (neu gibibyte)	2^{30}	1.073.741.824 Bytes
1 Terabyte (neu tebibyte)	2^{40}	1.099.511.627.776 Bytes
1 Petabyte (neu pebibyte)	2^{50}	1.125.899.906.842.624 Bytes

Informationsübertragung

- Information (v. lat.: in-form-are) = Kenntnis über irgendwas
- Austausch über **Nachrichten**

Nachricht	Zusammenstellung von Zeichen (Symbolen) zur Informationsübermittlung
Zeichen/Symbol	Element eines Zeichenvorrates festgelegter Menge (z.B. Alphabet)
Wort	Folge von Zeichen, die als Einheit betrachtet werden

Beispiel 1: Bei welcher Nachricht (a oder b) ist die Information größer?

a) Am 1. Juli war die Temperatur größer als 25 Grad.
b) Am 1. Juli betrug die Temperatur 29 Grad.

Informationsübertragung

- Information (v. lat.: in-form-are) = Kenntnis über irgendwas
- Austausch über **Nachrichten**

Nachricht	Zusammenstellung von Zeichen (Symbolen) zur Informationsübermittlung
Zeichen/Symbol	Element eines Zeichenvorrates festgelegter Menge (z.B. Alphabet)
Wort	Folge von Zeichen, die als Einheit betrachtet werden

Beispiel 2: Bei welcher Nachricht (c oder d) ist die Information größer?

a) Am 1. Juli war die Temperatur größer als 25 Grad.
b) Am 1. Juli betrug die Temperatur 29 Grad.

a) Am 1. Juli war die Temperatur größer als 25 Grad.
b) Am 1. Januar war die Temperatur größer als 25 Grad.

Informationsübertragung

Gehalt an Information unterscheiden sich je nach Empfänger
→ **Informationsgehalt einer Nachricht x**

Shannonsches Informationsmaß H: $I(x) = log_2 N \ bit$

mit N = Gesamtzahl der möglichen Zeichen $x_1 \dots x_n$
und gleicher Wahrscheinlichkeit des Auftretens von $x_1 \dots x_n$
mit $p(x_1) = p(x_2) = \dots p(x_n)$
und $p(x_1) + p(x_2) + \dots + p(x_n) = 1$

Beispiel 3: Zeichenvorrat: 10 Zeichen {0, 1, 2, 3, 4, 5, 6, 7,8 ,9}, die mit gleicher Wahrscheinlichkeit auftreten.

Der Informationsgehalt einer Nachricht, bestehend aus 1 Zeichen, ist somit:

$$I(x) =$$

Informationsübertragung

aus Beispiel 2: Bei welcher Nachricht (c oder d) ist die Information größer?

c) Am 1. Juli war die Temperatur größer als 25 Grad
d) Am 1. Januar war die Temperatur größer als 25 Grad.

- je seltener ein Zeichen x auftritt, desto größer sein **Informationsgehalt** $I(x)$
  genauer: $I(x) = \sim \frac{1}{p(x)}$
- für den Fall $p(x) = 1$ soll gelten: $I(x) = 0$
- es folgt: $I(x) = \log_2\left(\frac{1}{p(x)}\right)$

Informationsübertragung

aus Beispiel 2: Bei welcher Nachricht (c oder d) ist die Information größer?

c) Am 1. Juli war die Temperatur größer als 25 Grad
d) Am 1. Januar war die Temperatur größer als 25 Grad.

- je seltener ein Zeichen x auftritt, desto größer sein **Informationsgehalt** $I(x)$
  genauer: $I(x) = \sim \frac{1}{p(x)}$
- für den Fall $p(x) = 1$ soll gelten: $I(x) = 0$
- es folgt: $I(x) = \log_2\left(\frac{1}{p(x)}\right)$

Zusammenfassung:

Information ≙ Maß der **Ungewissheit**, mit der ein Empfänger eine Nachricht erwarten kann.

1 bit ≙ Information, welche bei **gleicher** Wahrscheinlichkeit zweier Alternativen durch die Kenntnis einer Alternative vermittelt wird.

Maschinen und Sprachen

Hardware - die Maschine

- Bestandteile eines EDV-Systems, i. A. Computer
- materielle bzw. technische Komponenten

Software - die Programmausstattung

- Bearbeitungsvorschriften für Hardware
- Unterteilung Betriebssystem - Anwendersoftware

Programmieren

- Umsetzung **Algorithmus** in Computerprogramm
- Verwendung einer Programmiersprache

Programmiersprache

- Ausdrucksmittel Menschen -> Maschine
- klare Definition durch **Syntax** und **Semantik**
- Programmiersprache erlernen vs. Programmieren erlernen

Maschinensprache

- **Maschinensprache** = Satz von Befehlen für eine ganz bestimmte Rechenmaschine

- Programmieren in Maschinensprache ist mühsam und fehleranfällig

```
LOAD    b,R0
LOAD    c,R1
ADD     R0,R1
SUB     2,R1
STORE   R1,a
```

Höhere Programmiersprache

```
a = b + c - 2;
```

- Orientierung an **Problemen statt Maschinen**

- Vorteile:
  - schnellere und sichere Programmerstellung
  - bessere Lesbarkeit
  - Wiederverwendbarkeit (Portierbarkeit)

- Nachteil:
  - Programme können nicht unmittelbar ausgeführt werden

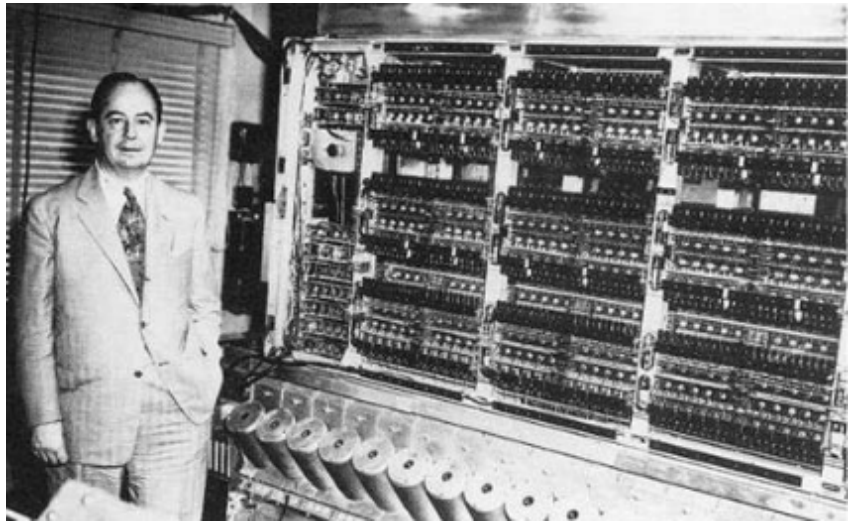
  → Übersetzung notwendig (**Interpreter** und **Compiler**)

Hardware: Die Maschine

- John **Von-Neumann-Modell** (1947)
- General Purpose Computer (allzwecktauglich)
- EVA-Prinzip (Eingabe, Verarbeitung, Ausgabe)
- nicht selbstständig arbeitsfähig, Anwendungsmaschine durch Software
- Prinzipien

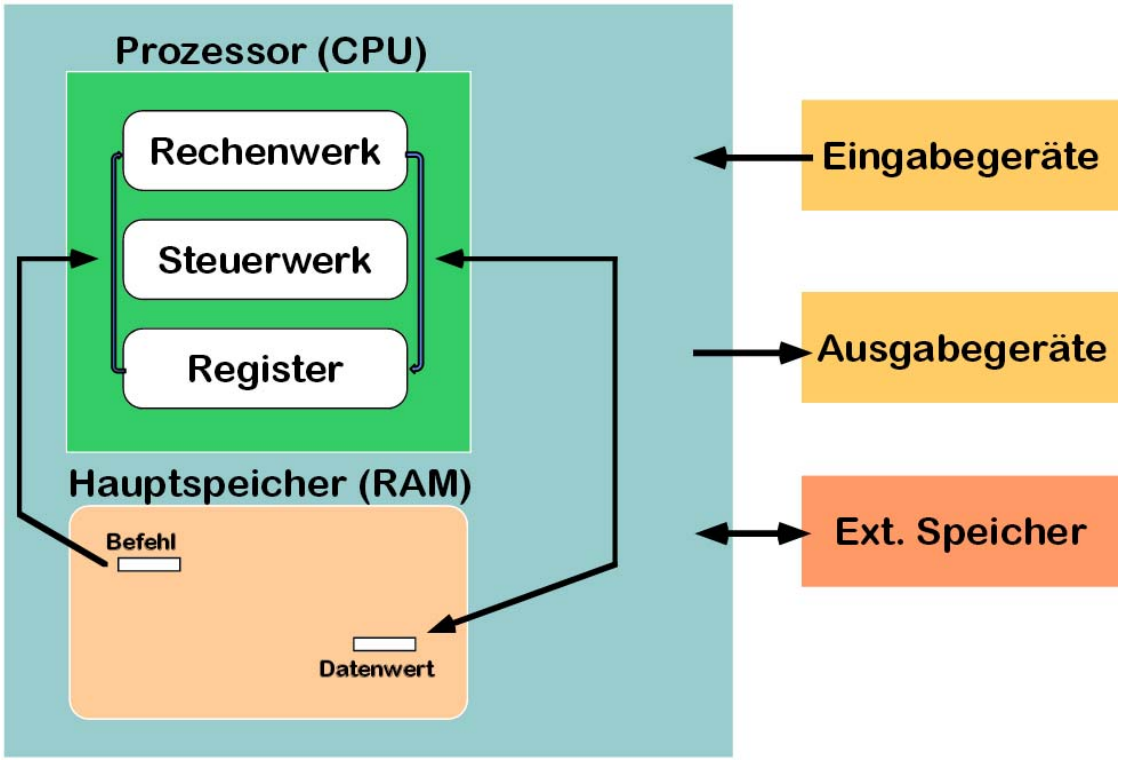
  Einfachheit
  (übersichtlich, minimaler HW-Aufwand)

  Hohe Flexibilität
  (bei genügend elementaren Befehlen)



Registermaschine auf Basis des von-Neumann-Modells

- Befehle und Daten im gleichen Speicher, Register als Zwischenspeicher im Prozessor
- sequentielle Verarbeitung von Befehl und Datum
- einzelner Verbindungsweg zwischen CPU ↔ Speicher

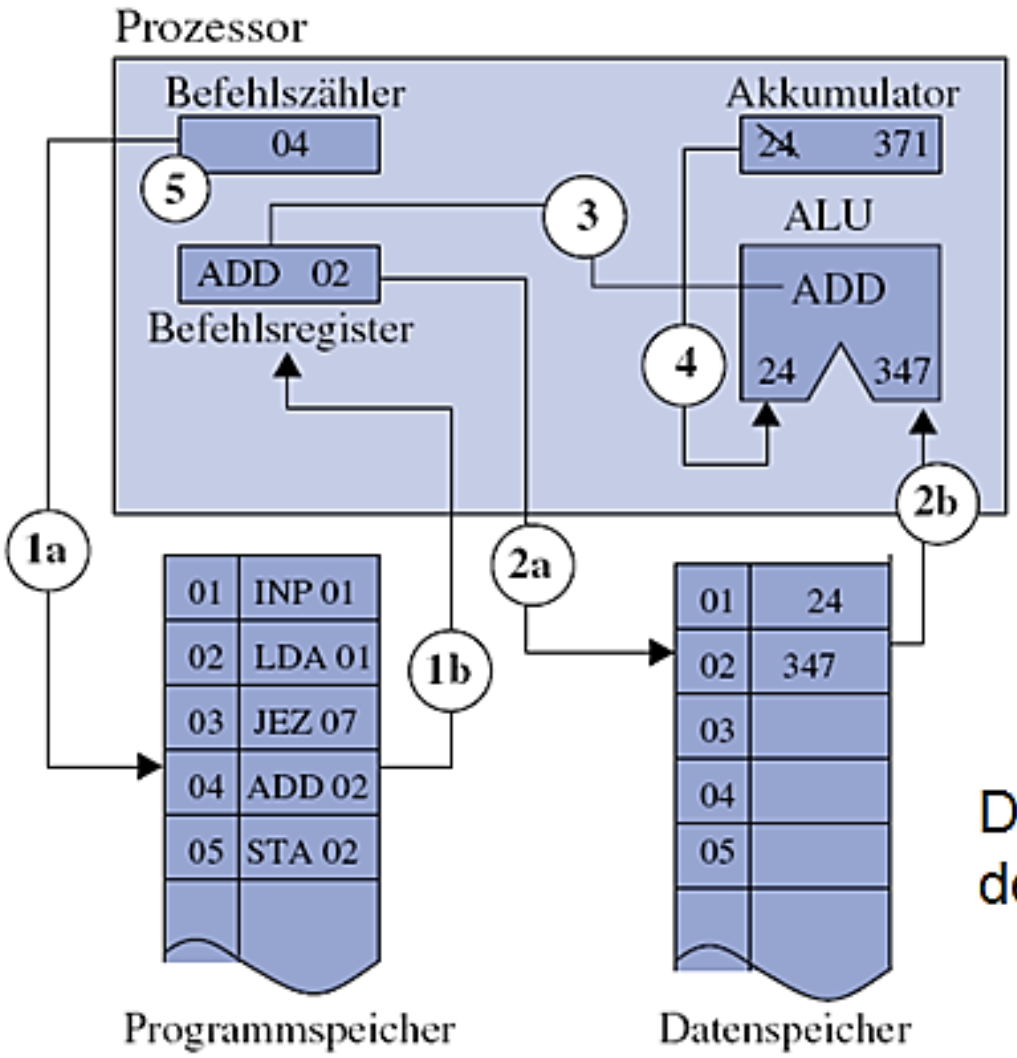


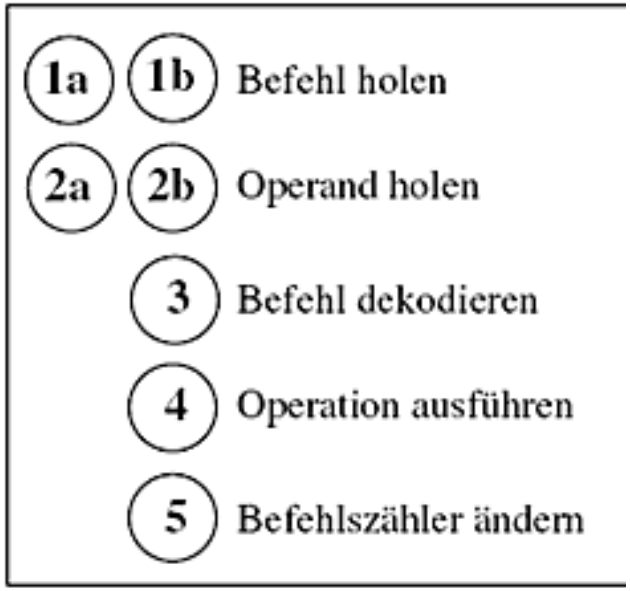
Beispiel: Programm "Summe"

Aufgabe: Summe von einzugebenden Zahlen ermitteln und ausgeben,
Abbruch bei Eingabe von "0"

```
01  INP  01     ; Zahl einlesen und an Adr. 1 (Datenspeicher) speichern
02  LDA 01      ; Lade Zahl von Adr. 1 (Datenspeicher) in Akkumulator
03  JEZ  07     ; Falls  Akku == 0, springe an Programmadr. 7 (OUT 02)
04  ADD 02      ; Addiere auf Akku Inhalt von Adresse 2 (Datenspeicher)
05  STA 02      ; Speichere Akku an Adresse 2 (Datenspeicher)
06  JMP 01      ; Springe zurück an Programmadr. 1 (INP 01)
07  OUT 02      ; Gib Inhalt von Adresse 2 (Datenspeicher) aus
08  HLT 99      ; Programmende
```

Programmstart
  Zahl einlesen
  Wenn Zahl = 0, springe zur Ergebnisausgabe
  Zahl auf bisherige Summe aufaddieren
  Springe zum Programmstart
  Ergebnis ausgeben
Programmende





Der fünfschrittige Befehlszyklus der Registermaschine

Von-Neumann-Rechner/Assembler Simulatoren

Johnny - A Simulator of a Simple von-Neumann Computer
http://sourceforge.net/projects/johnnysimulator/

Von-Neumann-CPU-Simulator
http://homepge.ruhr-uni-bochum.de/Daniel.Reinert/Software/von_Neumann.htm

AssemberSim
http://www.assemblersim.de/

GNUSim8085
http://www.gnusim8085.org

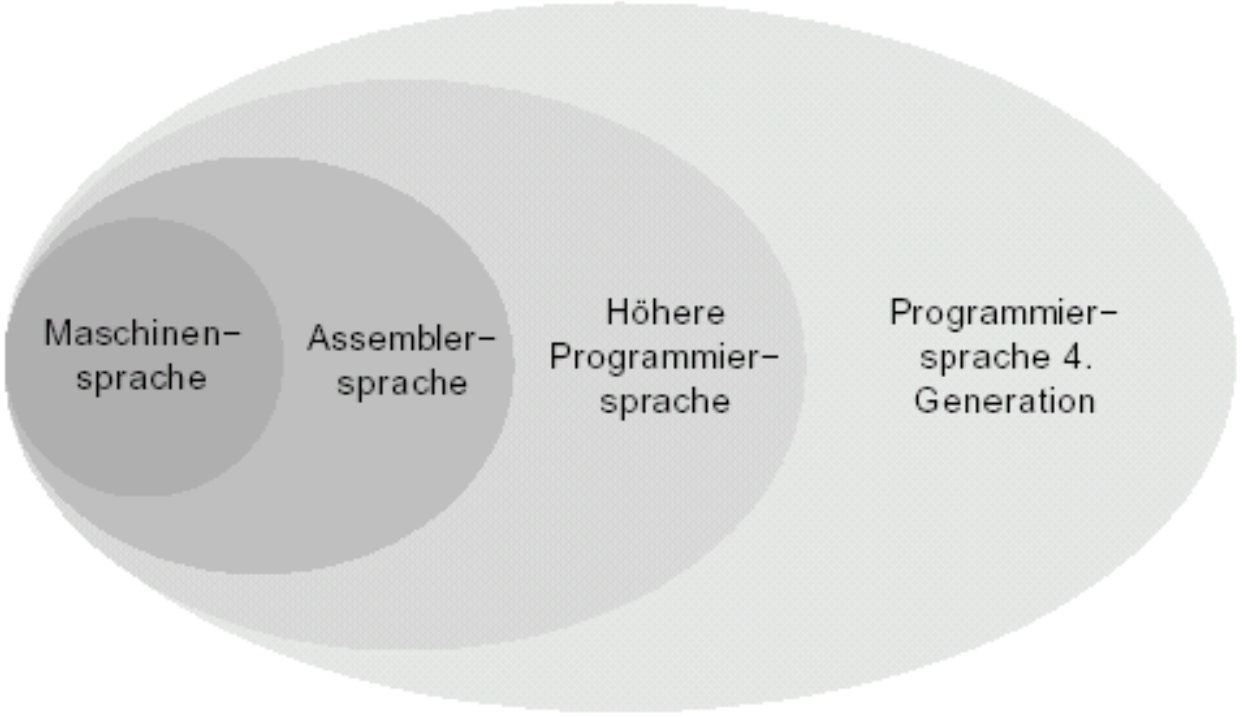
Beispiel: Programm "Summe"

Aufgabe: Summe von einzugebenden Zahlen ermitteln und ausgeben,
Abbruch bei Eingabe von "0"

```
01  INP  01     ; Zahl einlesen und an Adr. 1 (Datenspeicher) speichern
02  LDA 01      ; Lade Zahl von Adr. 1 (Datenspeicher) in Akkumulator
03  JEZ  07     ; Falls  Akku == 0, springe an Programmadr. 7 (OUT 02)
04  ADD 02      ; Addiere auf Akku Inhalt von Adresse 2 (Datenspeicher)
05  STA 02      ; Speichere Akku an Adresse 2 (Datenspeicher)
06  JMP 01      ; Springe zurück an Programmadr. 1 (INP 01)
07  OUT 02      ; Gib Inhalt von Adresse 2 (Datenspeicher) aus
08  HLT 99      ; Programmende
```

Programmstart
  Zahl einlesen
  Wenn Zahl = 0, springe zur Ergebnisausgabe
  Zahl auf bisherige Summe aufaddieren
  Springe zum Programmstart
  Ergebnis ausgeben
Programmende

Maschinensprache – Höhere Programmiersprache



Sprachevolution ⟶

Imperative Programmiersprachen

- Sprachen der ersten bis dritten Generation

- Beschreibung von Programmabläufen (Funktionsorientierte Methode)

1) Problemlösung = Folge von Einzelschritten (-> **Algorithmus**)

```
BeginneProgramm

        Anweisung 1
        Anweisung 2
        Anweisung 3
        ...
BeendeProgramm
```

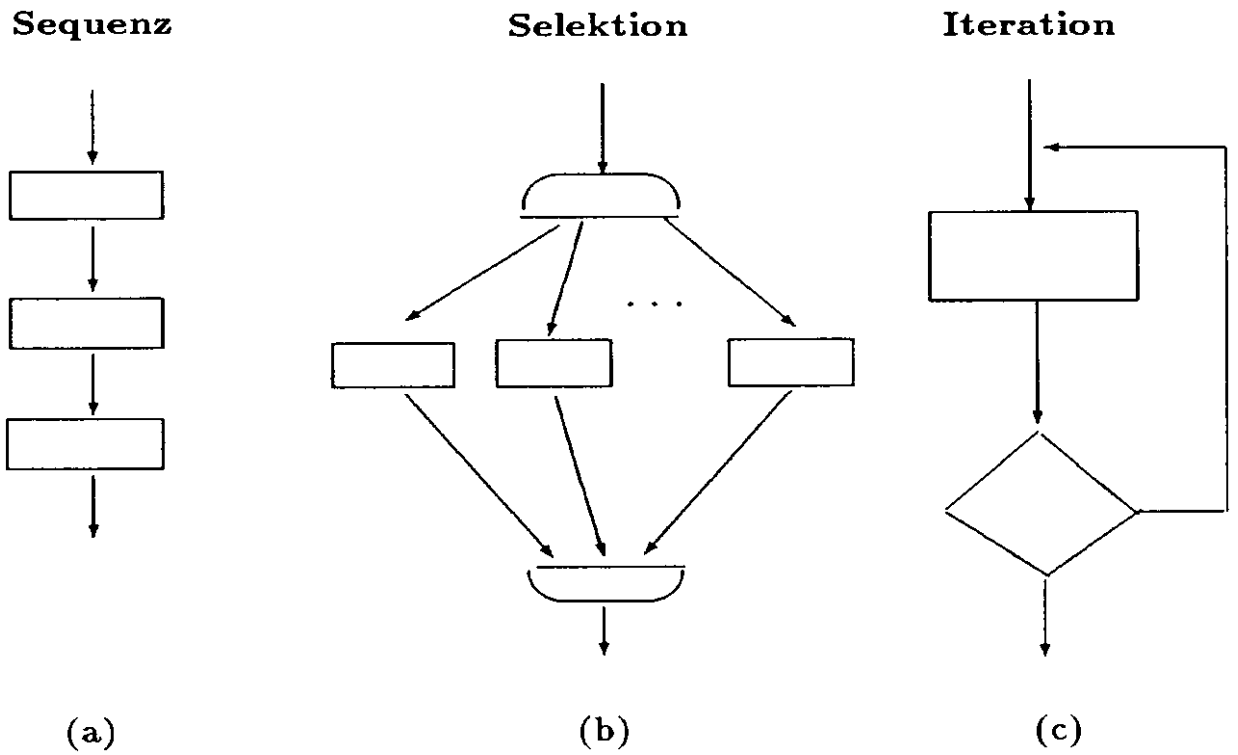
2) Programm = Vereinbarungen und Verarbeitungen (-> **Algorithmus**)

3) Anwendung des **Prinzips der Strukturierten Programmierung**

- sprachenübergreifendes Programmierparadigma
- Methoden und Regeln zum Entwurf von Algorithmen/Programmen
- vollständige Aufteilung des Problems in Teilprobleme
- Herabsetzung der Fehleranfälligkeit

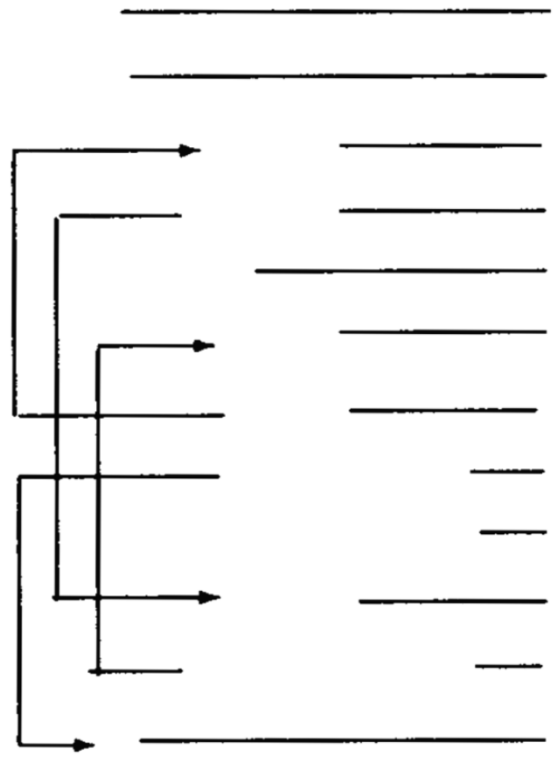
Prinzipien der Strukturierten Programmierung (1)

- Struktur-Theorem (Edsger W. Dijkstra 1972)
- Hierarchische Programmorganisation (Trennung **Steuerung** - **Verarbeitung**)
- Beschränkung der Ablauf-Steuerung auf 3 Grundelemente:



Prinzipien der Strukturierten Programmierung (2)

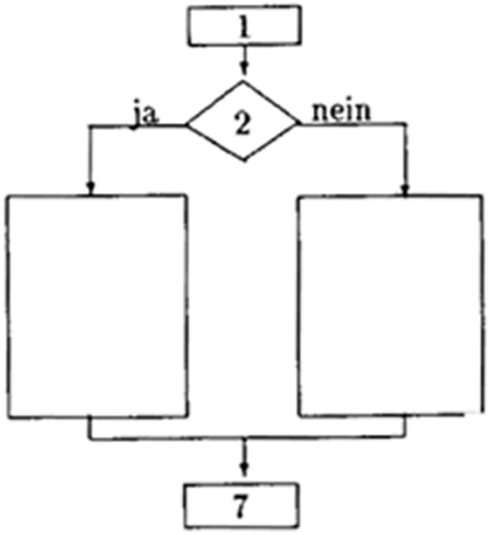
- Verzicht auf Sprünge ("GOTO")

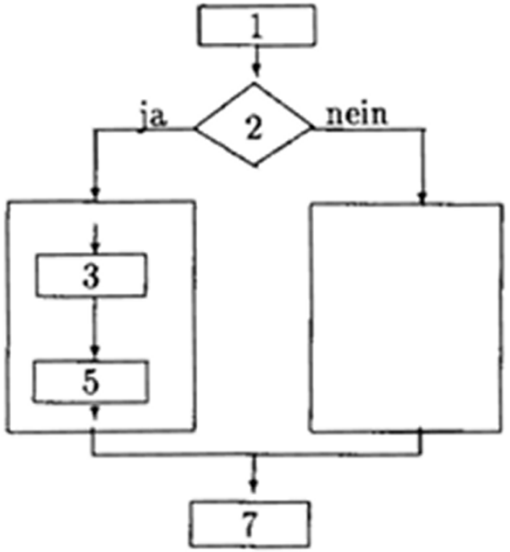


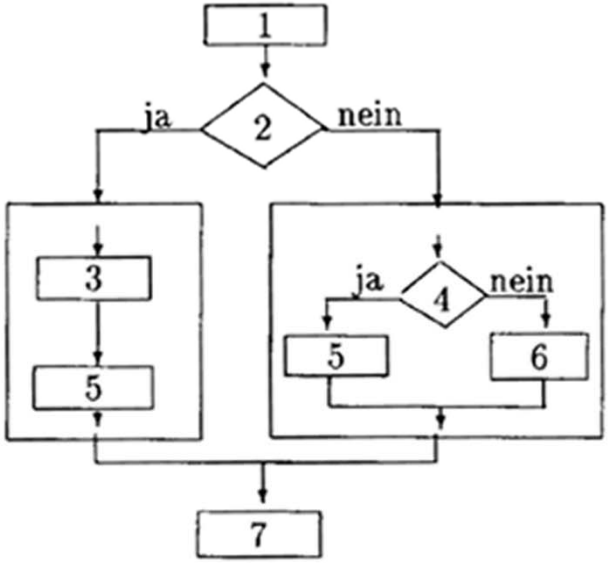
abstrahiertes Programmbeispiel

Prinzipien der Strukturierten Programmierung (3)

- Vorgehensweise: Top-Down Zerlegung, schrittweise Verfeinerung



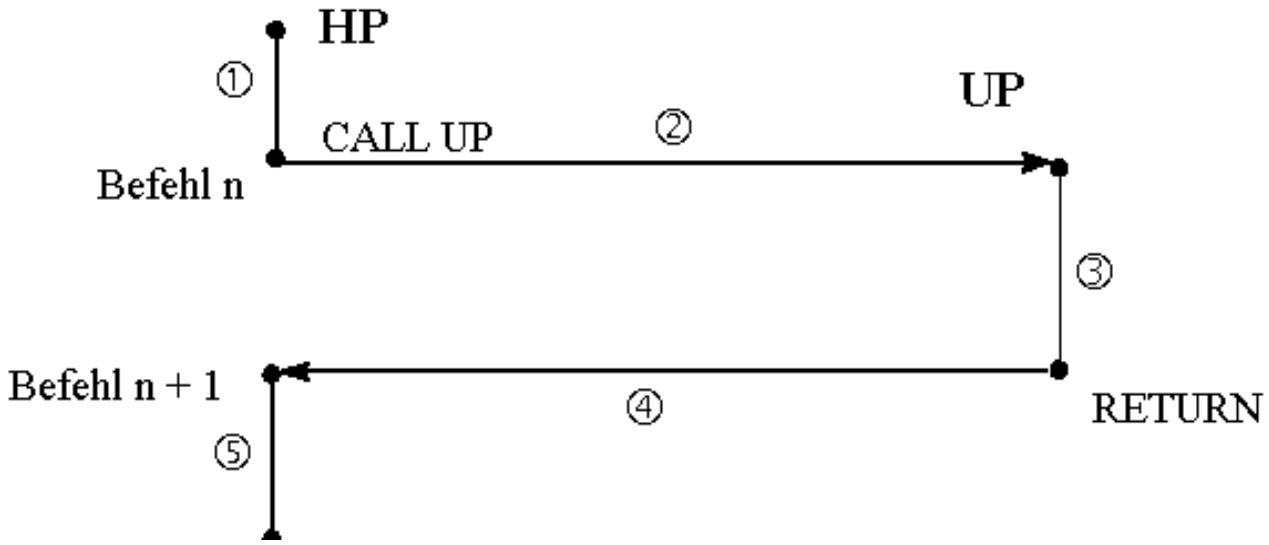




Prozedurale Programmierung

Prozedur/Unterprogramm: an anderer Stelle festgelegte Menge von Operationen

- Informationsaustausch mittels Parameterübergabe Ergebnisrückgabe
- Strukturierung bei wiederholter Verwendung von Programmteilen (mit gleichen oder veränderlichen Parametern)



Algorithmen

Algorithmus = Lösungsverfahren

Menge von Regeln für ein **Verfahren**, um aus gewissen **Eingabegrößen** bestimmte **Ausgabegrößen** herzuleiten

*"Der Algorithmus ist als **Oberbegriff zu Programm** zu betrachten, bei dem im Gegensatz zu einem Programm die strengen syntaktischen Regeln der Programmiersprache nicht eingehalten zu werden brauchen"*

aus "Programmieren in C" R.Klima/S.Selberherr

"Wenn ein Algorithmus für ein Problem existiert, dann ist das Problem durch Computer lösbar."

"Dixit algorismi" (= "so sprach Al Khowarismi")

Al Khowarismi (783-850): "Regeln der Wiedereinsetzung und Reduktion"

Beispiel aus dem Alltag: "Gulaschsuppe zubereiten"

Zu manipulierende Objekte:

350 g Rindfleisch, 3 Zwiebeln, 50 g Schweineschmalz, 15 g Mehl, 1 kleine Dose Tomatenmark, 3/4 l Wasser, Salz, Paprika, Majoran

Hilfsobjekte:

Herd, Kochgeschirr

Anweisungen:

Fleisch und Zwiebeln würfeln
in Schmalz andünsten
mit Mehl bestäuben und kurz anrösten
Tomatenmark zugeben
mit Wasser auffüllen
garen
mit Salz, Paprika und Majoran abschmecken

Eigenschaften

- **Finitheit, endliche Vorschrift**
- **Effektivität**
- **Determiniertheit**
- **Determinismus**
- **Abstrahierung**
- **Terminierung**
- **Eingabespezifikation (Anforderungen)**
- **Ausgabespezifikation (Zusicherungen)**

Eigenschaften

- **Finitheit, endliche Vorschrift**
  Verfahren muss in einem endlichen Text vollständig beschreibbar sein
- **Effektivität**
  Jeder Schritt des Verfahrens muss effektiv (d.h. tatsächlich) mechanisch ausführbar sein.
- **Determiniertheit**
  Wiederholbarkeit bzgl. Ein- Ausgabedaten
- **Determinismus**
  vorherbestimmter Ablauf zu jedem Zeitpunkt
- **Abstrahierung**
  anwendbar auf eine Klasse von Problemen (steuerbar über Parameter)
- **Terminierung**
  liefert Ergebnis nach endlich vielen Schritten
- **Eingabespezifikation (Anforderungen)**
  genaue Spezifikation der erforderlichen Eingabegrößen
- **Ausgabespezifikation (Zusicherungen)**
  genaue Spezifikation der zugesicherten Ausgabegrößen bzw. Resultate

Typische und bekannte Algorithmen

- Euklidischer Algorithmus zur Bestimmung des ggT
- Algorithmen zur Bestimmung von Primzahlen
- Gaußscher Algorithmus zur Lösung eines LGS
- Suchen und Sortieren
- Bearbeiten bestimmter Datenstrukturen (z.B. Listen, Bäume, Graphen)
- Zeichenketten-Verarbeitung
- Mustererkennung
- Statistische Berechnungen

Collatz-Problem

Bildungsgesetz:

- Beginne mit irgendeiner natürlichen Zahl $n > 0$
- Ist n gerade, so nimm als nächstes $n/2$
- Ist n ungerade, so nimm als nächstes $3n + 1$
- Wiederhole die Vorgehensweise mit der erhaltenen Zahl

So erhält man zum Beispiel für die Startzahl 19 die Folge:

19, 58, 29, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, 4, 2, 1, …

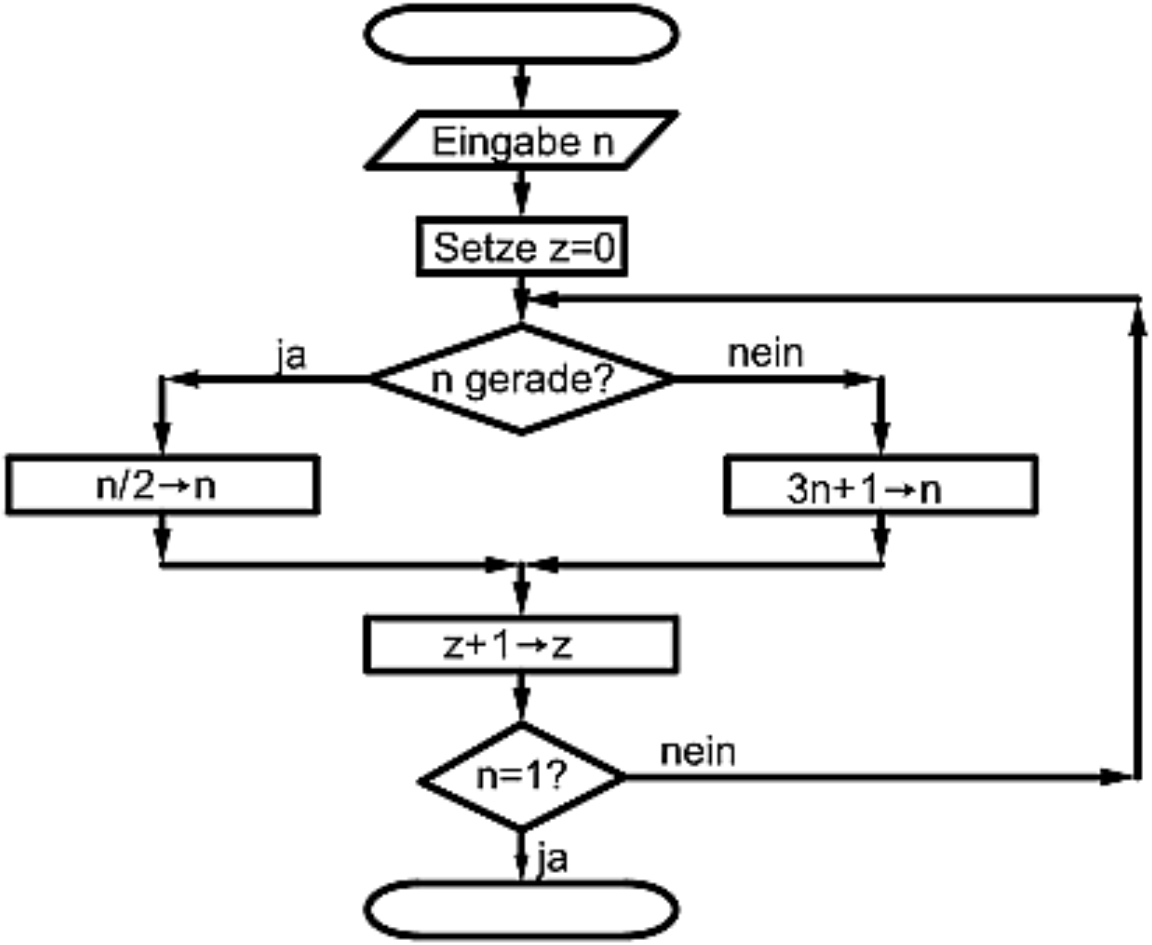
Collatz-Vermutung:

Jede so konstruierte Zahlenfolge mündet in den Zyklus 4, 2, 1,
egal, mit welcher natürlichen Zahl n man beginnt.

Möglichkeiten der Darstellung

- natürliche Sprache
- Pseudo-Code (formale Sprache)
- Programmablaufplan (PAP)
- Struktogramm
- Programmiersprache

Programmablaufplan Collatz-Problem



Programmablaufplan (PAP) nach DIN66001 - Ausgewählte Symbole