

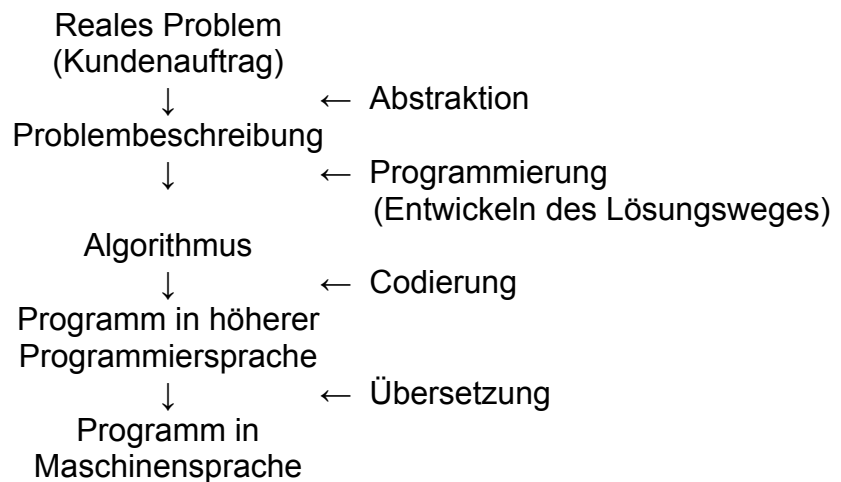
**Inhaltsverzeichnis:**

Wird bei Fertigstellung hinzugefügt!!

## Programmierung – Codierung

- Programmierung ist das Entwickeln von Lösungswegen für die Umsetzung auf einem EDV-System
  - Entwickeln von Algorithmen
  - Zerlegen eines Problems in Einzelschritte, die vom Computer abgearbeitet werden können
- Codierung ist die Umsetzung des Lösungsweges in eine Programmiersprache

## Vom Problem zum Programm

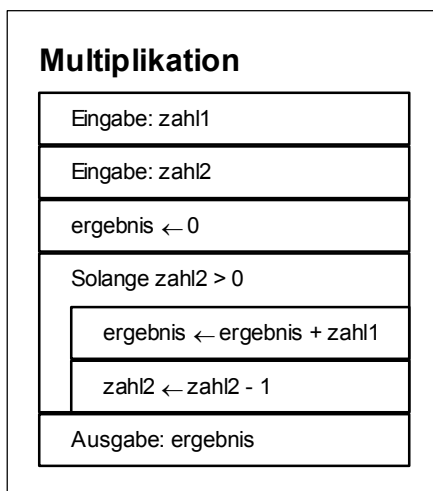


## Algorithmus

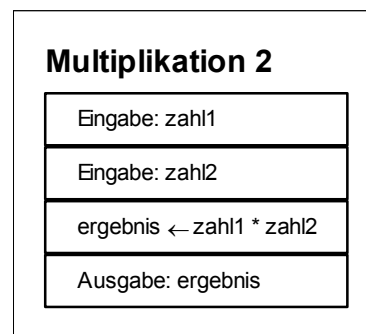
- Verhaltensmuster ohne direkten Bezug zu einem konkreten Computersystem
- logische Abfolge von Befehlen
- 3 Voraussetzungen
  - Ablauf / Reihenfolge der einzelnen Schritte ist fest vorgeschrieben
  - Alle Schritte müssen ausführbar sein
  - Das Verfahren führt immer zu einem Ende

## Beispiel für einen Algorithmus (Script Seite 3)

Multiplikation zweier Zahlen durch fortgesetzte Addition



vereinfacht:



### Aufgaben des Programmierers

- Unter Berücksichtigung der Programmvorgaben einen logischen Lösungsweg finden
  - Zerlegung der Aufgabe in Einzelschritte
  - Übertragung des Lösungsweges in eine Programmiersprache
  - Testen des Programms auf logische Fehler
  - Abschließende Prüfung des Programms hinsichtlich der Vorgaben
  - Dokumentieren des Programms
    - Anwender-Doku (Benutzerhandbuch)
    - Entwickler-Doku

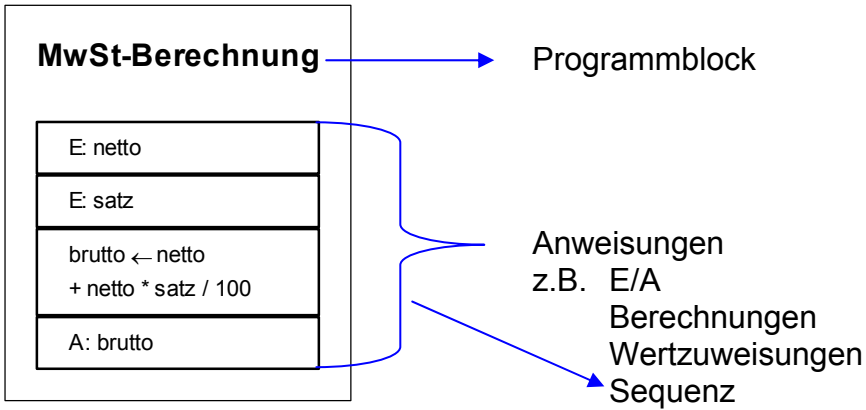
### Kurze Einführung in die Programmiersprachen

1. Generation → Maschinensprachen
  - binäre Form
2. Generation → Assemblersprachen
  - Verwendung von Symbolworten
3. Generation → prozedurale, problemorientierte Sprachen z.B. BASIC, Visual Basic, C, COBOL, PASCAL, etc.
4. Generation → nichtprozedural, nicht mehr beschreibend
  - deskriptiv
5. Generation → Künstliche Intelligenz z.B. LISP
  - Versuch menschliche Denkweise auf den Computer zu übertragen

## Grundelemente der Programmierung

- ① **Eingabe:** Erfassung von Daten über Eingabegeräte
- ② **Ausgabe:** Bildschirm, Drucker, Datei, etc.
- ③ **Datentypen:** Festlegung welche Werte in Variablen und Konstanten gespeichert werden können
- ④ **Operationen / Operatoren:**
  - Berechnungen, Vergleiche, Wertzuweisungen
  - Operatoren
    - Arithmetische Operatoren
      - + - \* / MOD DIV (\)
      - MOD → Modulo-Division
      - Liefert den ganzzahligen Rest einer Division
      - z.B.  $8 / 4 = 2$                        $7 / 3 = 2R1$
      - $8 \text{ MOD } 4 = 0$                  $7 \text{ MOD } 3 = 1$
    - DIV → Integer-Division
    - liefert das ganzzahlige Ergebnis einer Division ohne Rest
    - z.B.  $8 \text{ DIV } 4 = 2$                  $7 \text{ DIV } 3 = 2$
  - relationale Operatoren
    - > größer
    - < kleiner
    - = gleich
    - <> ungleich bzw. kleiner oder größer
    - >= größer oder gleich
    - <= kleiner oder gleich
  - logische Operatoren
    - AND, OR, NOT
    - Verknüpfungsoperatoren
    - Punkt vor Strich → AND vor OR → NOT geht vor alle
- ⑤ **Bedingte Ausführung**
  - Wenn... Dann... Sonst...
  - Weiterer Programmverlauf hängt von einer Bedingung ab
- ⑥ **Schleifen / Wiederholungen**
  - Bestimmter Programmteil wird wiederholt ausgeführt
  - Anzahl der Wiederholungen hängt von einer Bedingung ab
- ⑦ **Unterprogramme**
  - Teile des Programms werden ausgelagert
  - Vorteile:
    - Übersichtlichkeit
    - einfachere Programmierung / Codierung
    - leichtere Fehlersuche

**Symbole des Struktogramms DIN 66261 nach Nassi-Shneiderman**



Eingaben:

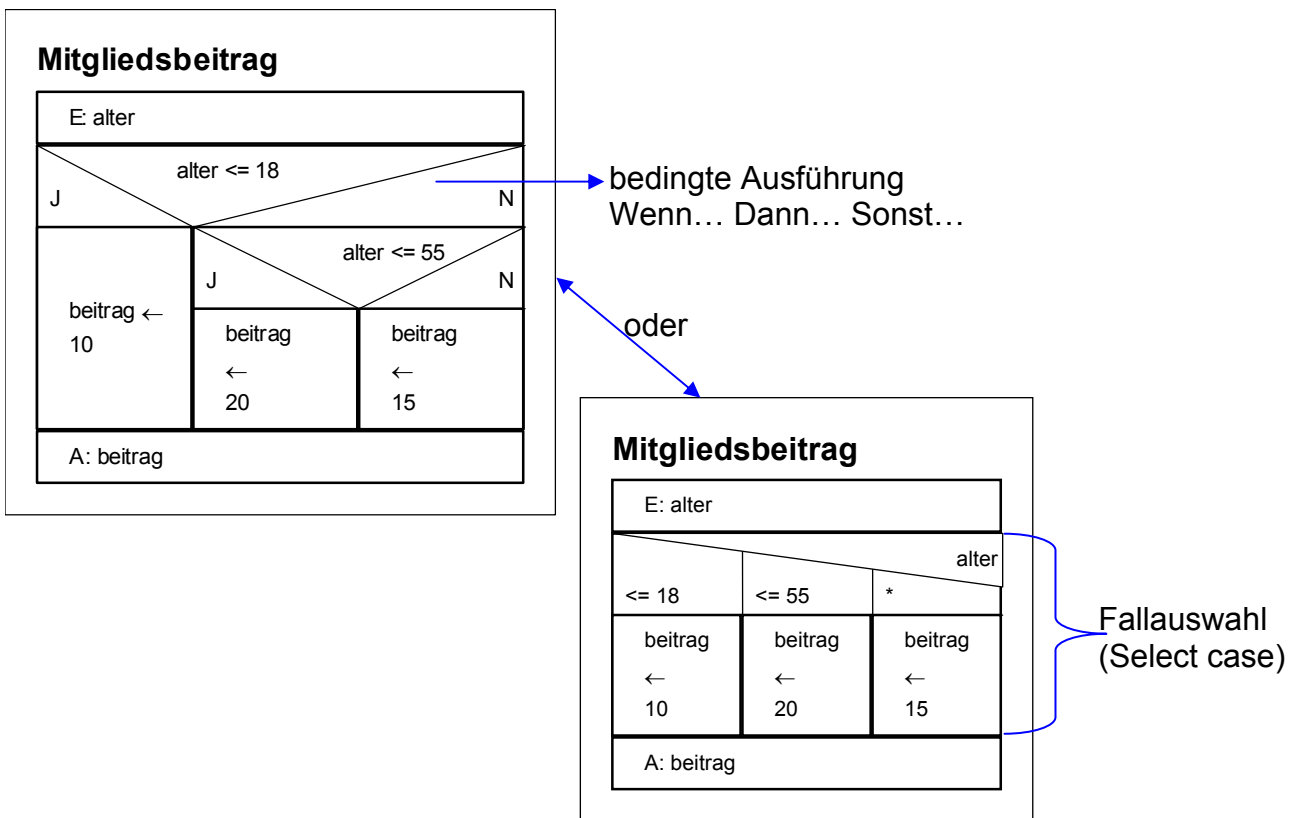
E: netto  
Eingabe: netto  
con → netto

Ausgaben:

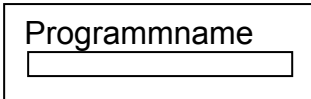
A: brutto  
Ausgabe: brutto  
BS ← brutto

Aufgabe:

Eingabe eines Alters  
Ausgabe des Mitgliedsbeitrags  
Verarbeitungshinweise:  
Alter bis 18 Jahre Beitrag € 10  
19 bis 55 Jahre Beitrag € 20  
ab 56 Jahre Beitrag € 15



Unterprogrammssymbol:



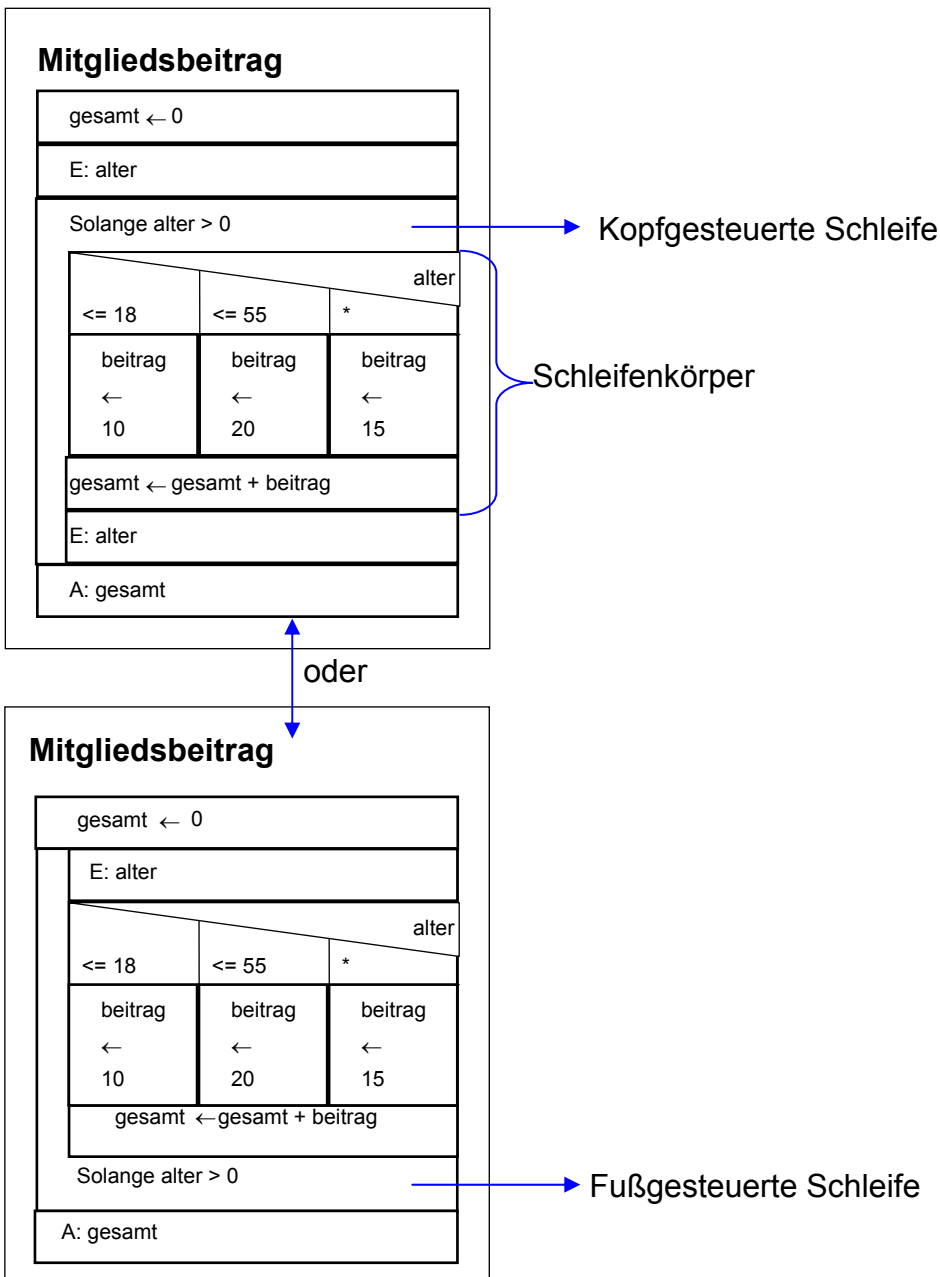
Aufgabe:

Eingabe des Alters

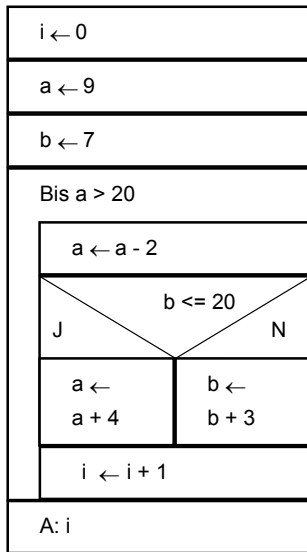
Ausgabe: Gesamtbeitrag

Beitragsberechnung siehe vorherige Aufgabe

Abbruch der Eingabe, wenn Alter = 0



### Schleifen-Beispiel

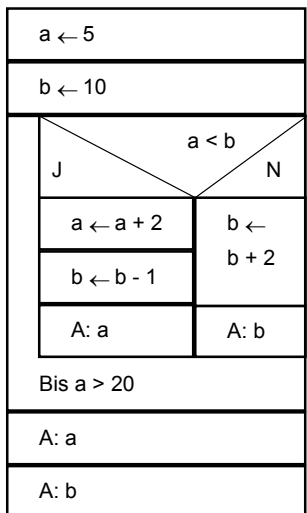


$i = ?$

$i = 6$

i	a	b
0	9	7
1	7	9
2	11	13
3	9	11
4	13	15
5	11	13
6	15	17
7	13	15
8	17	19
9	15	17
10	19	21

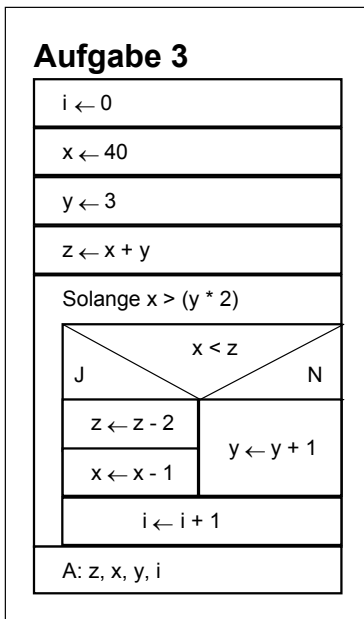
### Aufgabe 2



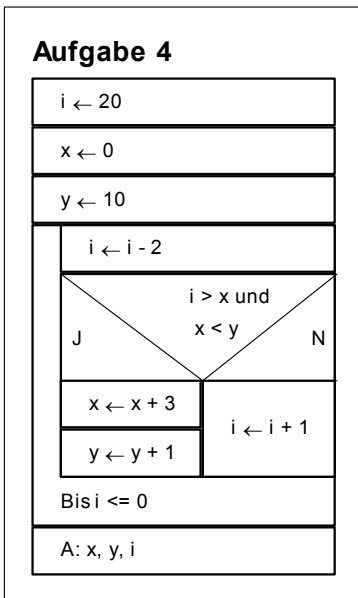
$a = ?$   
 $b = ?$

$a = 21$   
 $b = 20$

a	b
5	10
7	9
9	8
11	10
13	9
15	11
17	13
19	12
21	14
23	13
25	15
27	17
29	16
31	18
33	17
35	19
37	21
39	21
41	20

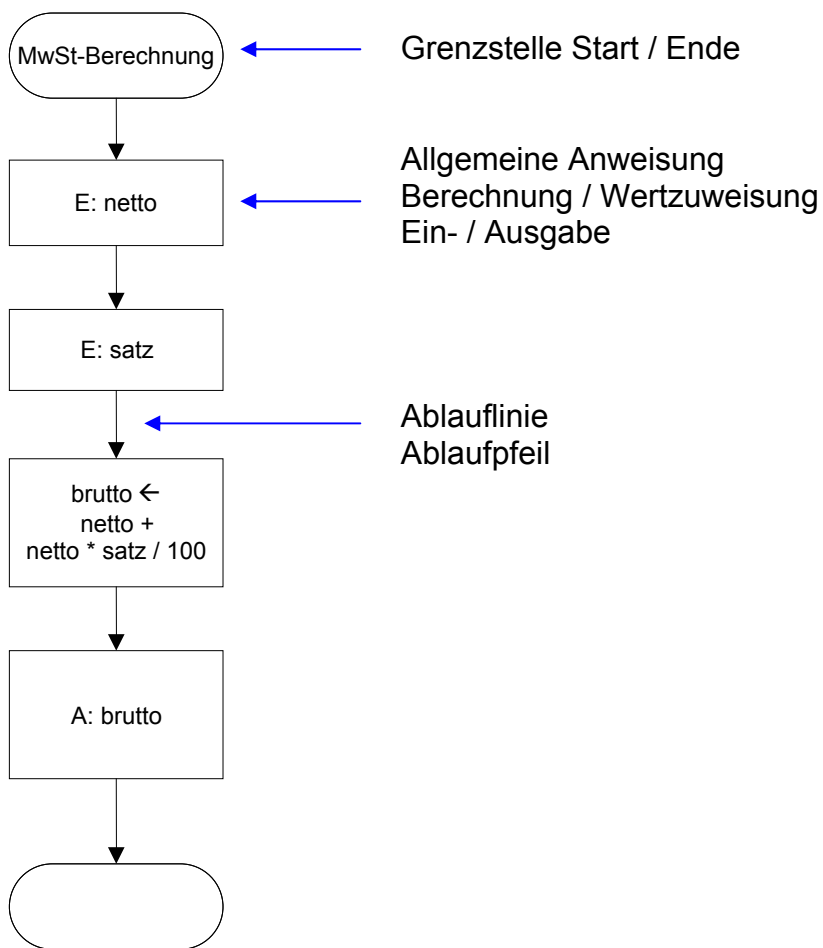


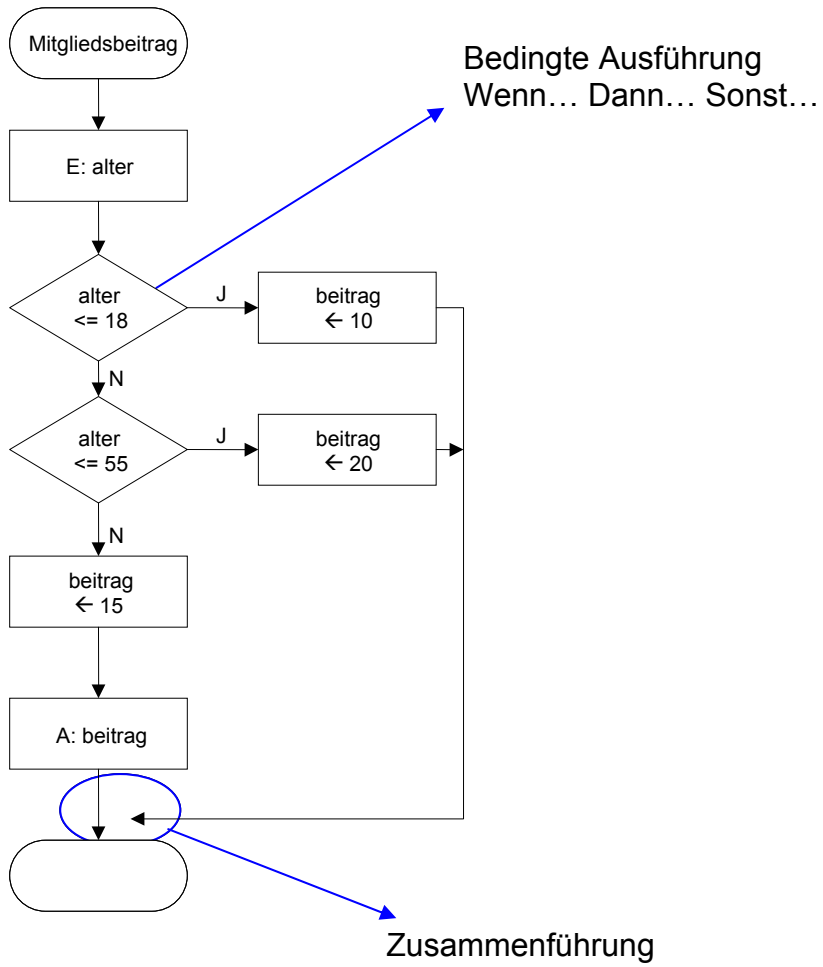
i	x	y	z
0	40	3	43
1	39	3	41
2	38	3	39
3	37	3	37
		6	
4		4	
		8	
5		5	
		10	
6		6	
		12	
7		7	
		14	
8		8	
		16	
9		9	
		18	
10		10	
		20	
11		11	
		22	
12		12	
		24	
13		13	
		26	
14		14	
		28	
15		15	
		30	
16		16	
		32	
17		17	
		34	
18		18	
		36	
19		19	
		38	



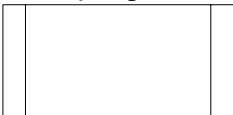
i	x	y
20	0	10
18	3	11
16	6	12
14	9	13
12	12	14
10		
11		
9		
10		
8		
9		
7		
8		
6		
7		
5		
6		
4		
5		
3		
4		
2		
3		
1		
2		
0		
1		
-1		
0		

**PAP-Symbole      Programm-Ablauf-Plan nach DIN 66001**

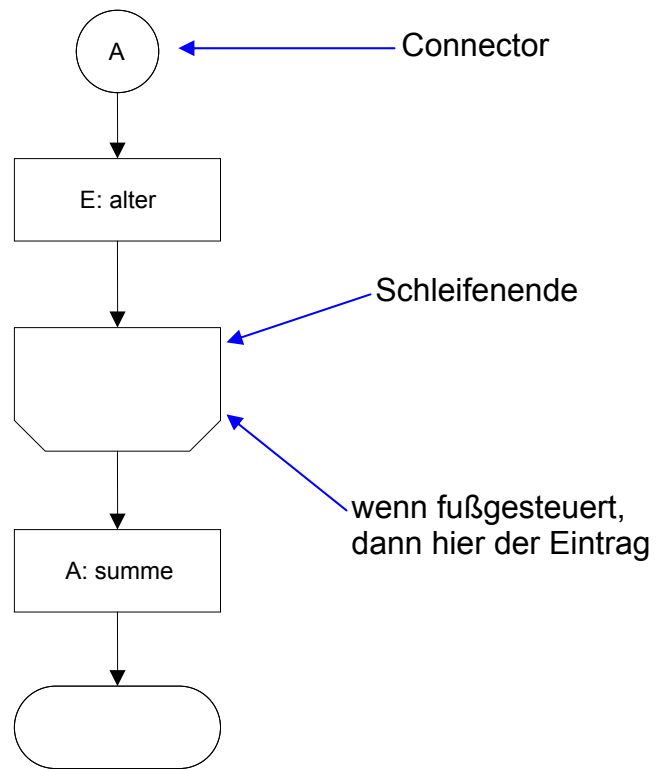
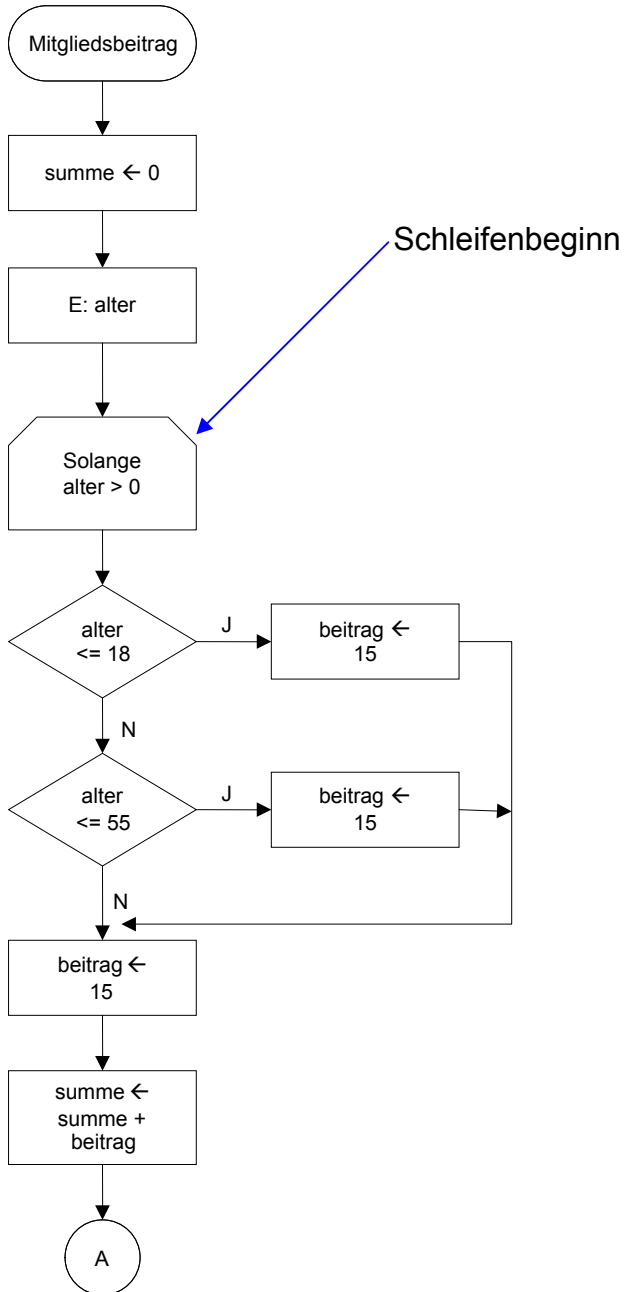




Unterprogrammssymbol:

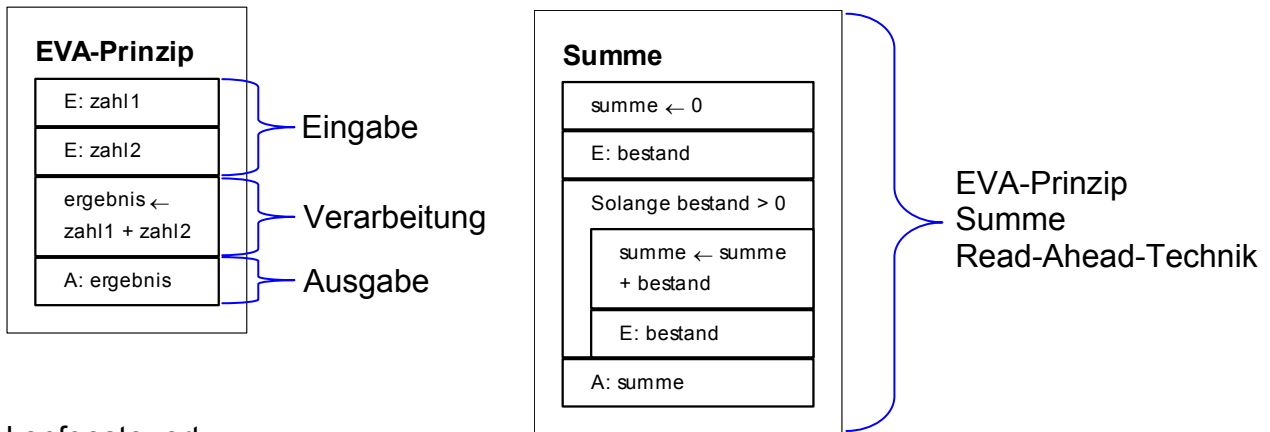


# kopfgesteuert

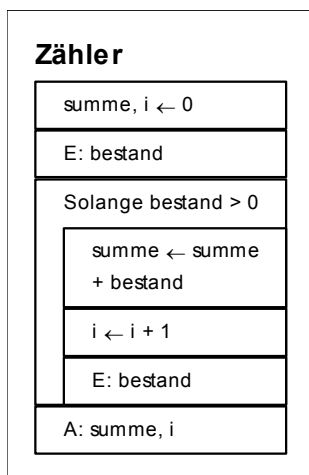


## Grundalgorithmen

- EVA-Prinzip
- Summe
- Zähler
- Durchschnitt
- Maximal-Wert
- Minimal-Wert
- Read-Ahead-Technik (vorauslesen)
- Plausibilitätsprüfung

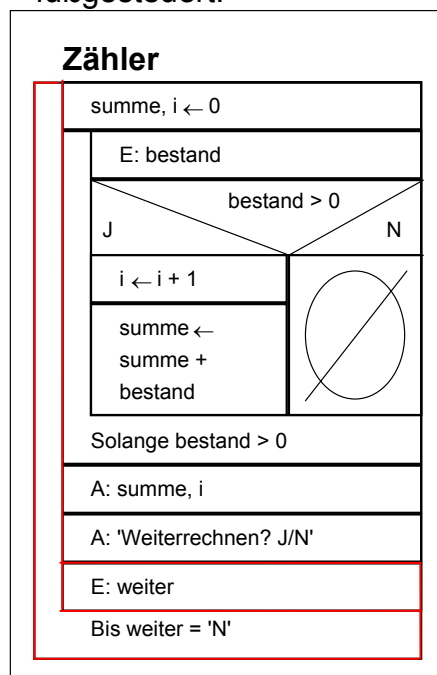


kopfgesteuert:



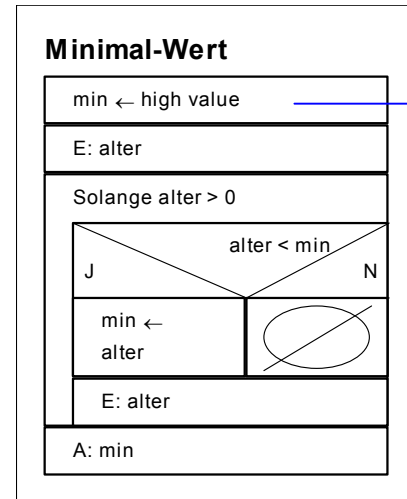
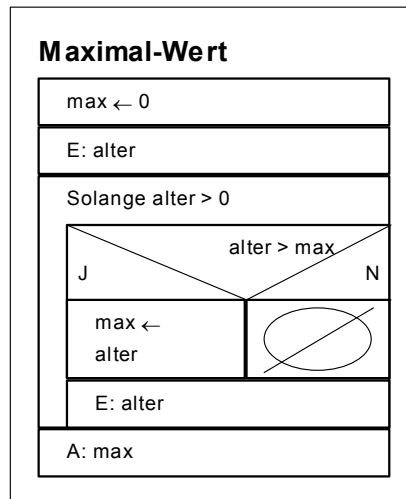
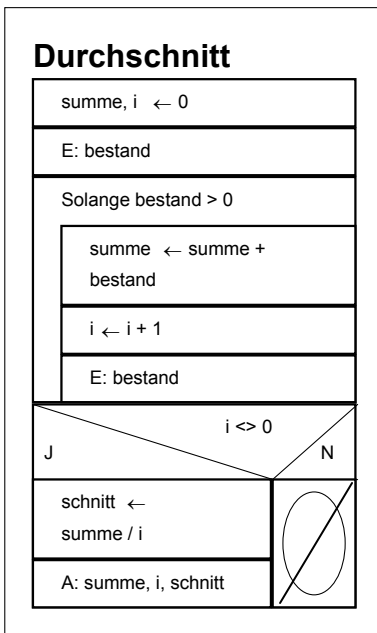
summe	i	bestand
0	0	200
200	1	100
300	2	50
350	3	100
450	4	0

fußgesteuert:



fußgesteuert wird wenigstens einmal durchlaufen

rot = mehrere Berechnungen sollten immer fußgesteuert sein



→ höchster Wert

wenn eine Variable auf beiden Seiten des Pfeils steht, dann muss die Variable vorher auf „0“ gesetzt werden

Bsp:

summe  $\leftarrow$  0

summe  $\leftarrow$  summe + i

### String-Verarbeitung (Zeichenketten-Verarbeitung)

Zeichenketten werden Zeichen für Zeichen verarbeitet.

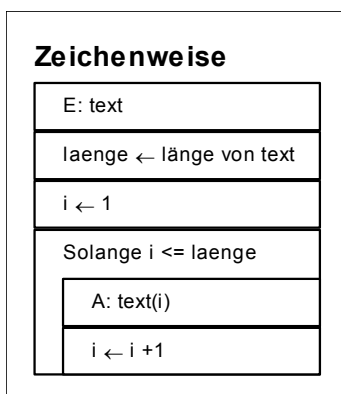
→ Kette wird zeichenweise gelesen

→ einzelne Zeichenposition wird durch einen Zähler ermittelt

Bsp.:

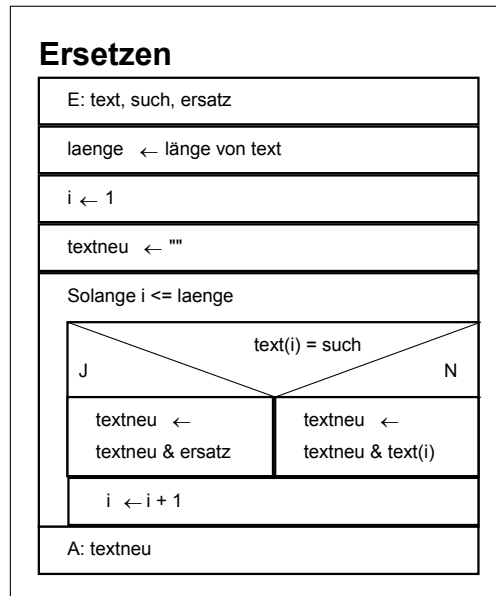
Eingabe eines Textes

Ausgabe: jeder einzelne Buchstabe (nacheinander)

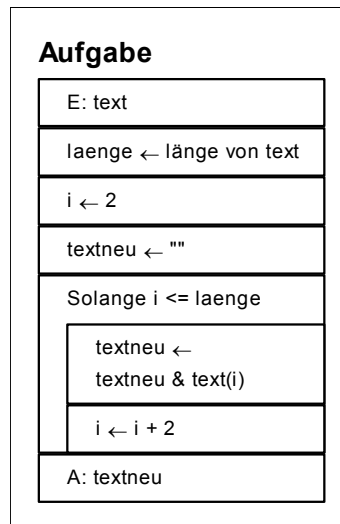


Bsp.:  
 Eingabe eines Textes  
 Eingabe eines Suchzeichens  
 Eingabe eines Ersatzzeichens  
 Ausgabe neuer Text

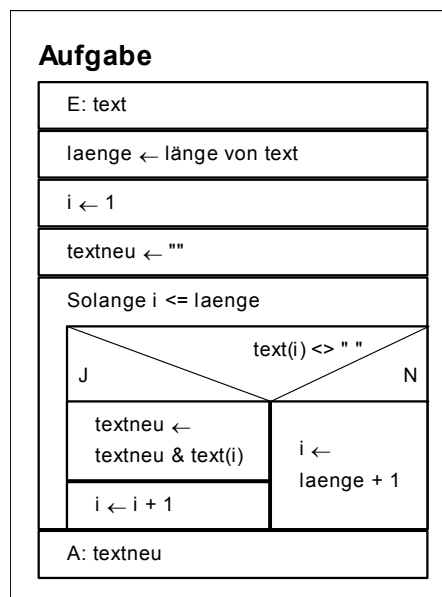
Beispiel:  
text            such            ersatz  
 Augsburg      u                      x  
 Ausgabe: Augsburg



Aufgabe:  
 Eingabe: text  
 Ausgabe: jedes zweite Zeichen  
 Bsp.: Augsburg  
 Ausgabe: „usug“



Aufgabe:  
 Eingabe eines Textes  
 Text enthält 1 Leerzeichen  
 Ausgabe: Text bis zum 1. Leerzeichen  
 Bsp.:  
 Eingabe: „Hans Huber“  
 Ausgabe: „Hans“



Aufgabe:

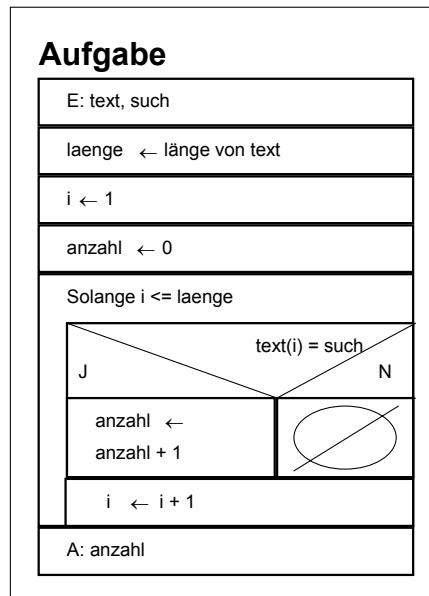
Eingabe: Text

Eingabe: Suchzeichen

Ausgabe: Anzahl Suchzeichen

Bsp.: „Augsburg“ such: „u“

Ausgabe: 2



**Nachtrag zu den Grundalgorithmen**

**Plausibilitätsprüfung**

Bsp.: siehe Aufgabe „Noten“

Verarbeitet werden nur korrekte Werte. Unter Umständen wird Eingabe wiederholt bis die Werte korrekt sind.

Bei Aufgabe „Noten“ waren nur die Noten 0 – 6 zu verarbeiten.  
(Note 0 als Abbruchkriterium)

**Sequentielle Dateiverarbeitung**

Sequentielle Dateien bestehen aus einer Auswahl von Datensätzen, die alle den gleichen Aufbau (= Struktur) haben und nur einzel nacheinander verarbeitet werden können.

Bsp.: Personaldatei

P1 Huber Gustav Abt203

P2 Maier Anton Abt201

P3 Schulz Berta Abt203

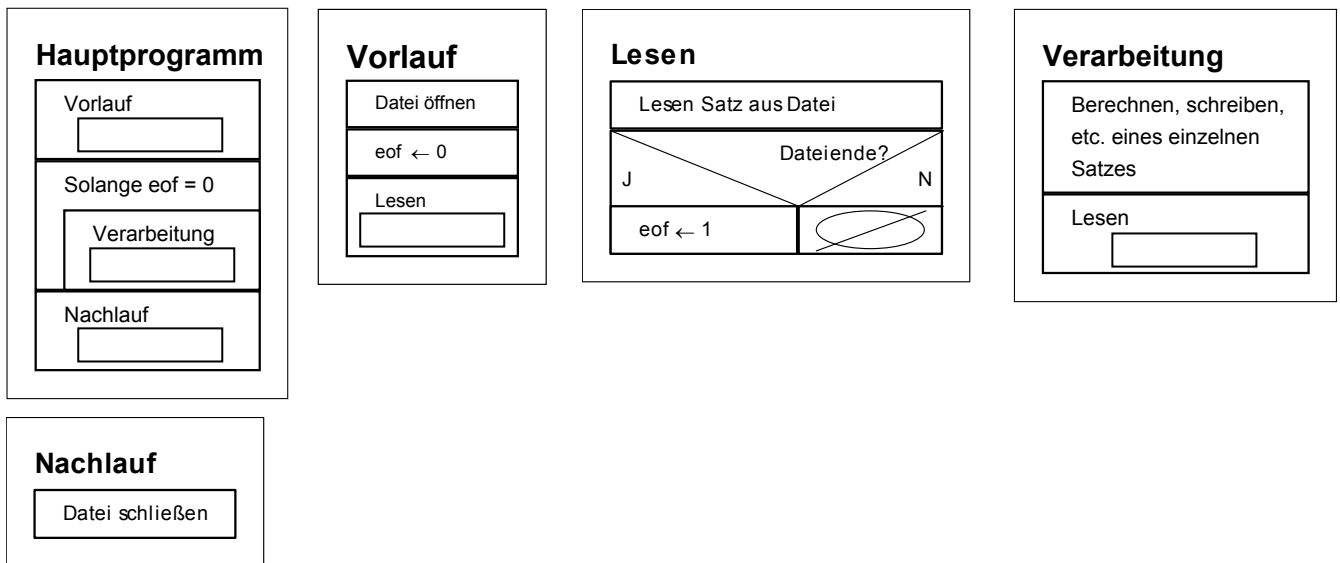
→ Dateiende (EOF – End of File)

↑  
dahinter Programmabsturz

## Grundsätzliche Vorgehensweise bei sequentieller Dateiverarbeitung

- ① Datei(en) öffnen für bestimmte Verarbeitungsart (Lesen, Schreiben) → 1x am Beginn  
↳ Vorlauf
  - ② Lesen des ersten Satzes und anschließend prüfen ob Dateiende gelesen wurde!
  - ③ Verarbeiten des gelesenen Satzes (z.B. Berechnen, drucken, Schreiben in eine andere Datei, etc.) } 1x pro Satz
  - ④ Lesen des nächsten Satzes mit Prüfung auf Dateiende! ↳ Verarbeitung in einer Schleife
  - ⑤ Evtl. Ausgaben von Gesamtergebnissen  
↳ Nachlauf
- 1x am Ende
- Wichtig: Datei(en) schließen

### Allgemeiner Ablauf



#### Aufgabe:

Personaldatei lesen. Prüfen ob AbtNr='A202'. Wenn ja dann Gehalt aufaddieren.

Außerdem den Satz in eine neue Datei schreiben.

Am Ende Gesamtgehalt am Bildschirm ausgeben.

Aufbau beider Dateien (Personal, PersA202):

PNR

Name

Vorname

Gehalt

AbtNr