2. Algorithmenbegriff

Keine Algorithmen: Anleitungen, Kochrezepte, Wegbeschreibungen, ...

<u>Algorithmus</u>:

Berechnungsvorschrift, die angibt, wie durch Ausführung bestimmter Elementaroperationen aus Eingabegrößen Ausgabewerte ermittelt werden.

Berechnungsvorschrift muß präzise und endlich sein, löst Problemenklasse

Elementaroperationen müssen von Maschine durchgeführt werden können

Abfolge der Schritte liegt fest, bzw. Wahlmöglichkeiten sind festgelegt

Beispiele:

Algorithmen zu Addition, Subtraktion, Division, Multiplikation, ...
euklidischer Algorithmus, ...

Ein Algorithmus?

Reifenauswechseln

1. Löse die Radmuttern.
2. Hebe den Wagen an.
3. Schraube die Radmuttern ab.
4. Tausche das Rad gegen ein anderes Rad aus.
5. Schraube die Radmuttern an.
6. Setze den Wagen ab.
7. Ziehe die Radmuttern fest.

Jeder Schritt definiert?

Festgelegte Folge von Ausführungsschritten?

Gibt es Elementaroperationen und *wer* führt sie aus?

Ein Nicht-Algorithmus

Wir machen Rührei
1. Erhitze Öl in der Pfanne bis sich Bläschen bilden.
2. Schlage die Eier in die Pfanne.
3. Füge eine Prise Salz hinzu und rühre gut durch.
4. Lasse alles brutzeln bis die Eier die richtige Konsistenz haben.

Euklidischer Algorithmus

Gesucht: größter gemeinsamer Teiler positiver ganzer Zahlen n, m

1. Wenn m kleiner als n, vertausche Werte von m und n.
2. Dividiere m durch n, nenne den Rest r.
3. Wenn r gleich 0 ist, so gib n aus und terminiere.
4. Wenn r nicht gleich 0 ist, so weise m den Wert von n zu und n den von r.
5. Gehe zu 2.

Jeder Schritt definiert?

Festgelegte Folge von Ausführungsschritten?

Gibt es Elementaroperationen und *wer* führt sie aus?

Eigenschaften von Algorithmen

- **Abstraktion: Lösung einer *Klasse* von Problemen**
- **Finitheit: Beschreibung endlich, jeweils nur endlich viel Speicherplatz**
- **Terminierung: wenn Algorithmen bei jeder Eingabe nach endlicher Zeit halten und ein Ergebnis liefern, so heißen sie *terminierend***
- **Determinismus: Algorithmus heißt deterministisch, wenn zu jeder Zeit der Ausführung nur *eine* Fortsetzungsmöglichkeit besteht**
- **Determiniertheit: Algorithmus heißt determiniert, wenn bei gleichen Eingabewerten stets das gleiche Ergebnis geliefert wird**

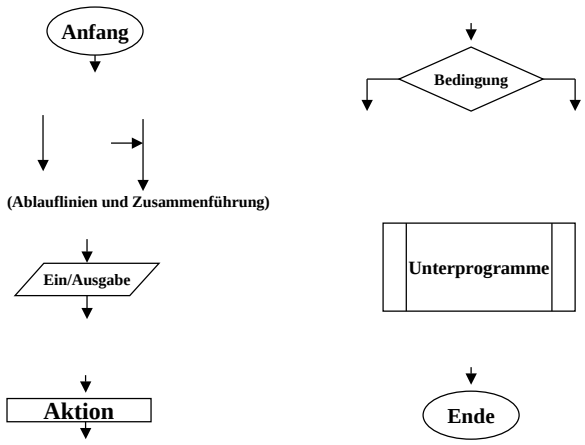
Deterministische Algorithmen immer determiniert, aber nicht umgekehrt!

Algorithmen sollten verständlich und effizient sein

Darstellung von Algorithmen

- **natürlichsprachlich**
  für die Kommunikation von Ideen oft ausreichend, muß präzisierbar sein
- **Stilisierte Prosa und Pseudo-Code**
  Mischung von Prosa und Konstrukten aus Programmiersprachen
- **Graphische Darstellungen, Ablaufpläne, Struktogramme**
  machen Kontrollfluß sichtbar
- **Formulierung in Programmiersprache**
  Präzision, Ausführbarkeit auf Maschine garantiert

Grafische Darstellungsmittel



Kontrollstrukturen und Ablaufsteuerung

Kontrollstrukturen bzw. Ablaufsteuerung legen die Reihenfolge und Bedingungen fest, unter denen jeder Algorithmusschritt ausgeführt wird. Wir betrachten explizit die Kontrollstrukturen Folge, Selektion, Iteration und Rekursion.

Folge (Sequenz)

Einfachste Form eines Algorithmus der in einer linearen Abfolge von genau beschriebenen Schritten besteht.

- Es wird immer nur ein Schritt ausgeführt.
- Jeder Schritt genau nur einmal ausgeführt.
- Strikt sequentielle Abarbeitung.
- Stop mit Abarbeitung des letzten Schrittes.

Seien A1, A2, ... Anweisungen. Eine Sequenz wäre z.B.

A26; A3; A15; ...

Für fast alle Aufgabenstellungen zu starr. Keine Alternativen erlaubt.

Selektion

Erlaubt Alternativen. Entscheidung der Alternative von Bedingung anbhängig. Allgemeine Form:

if (Bedingung) Anweisung
oder auch
if (Bedingung) Anweisung_1 **else** Anweisung_2

Beispiel.

```
int i;
...
if (i < 3) {i = 0;} else {i = 3;}
```

Jede Anweisung kann eine Folge von Anweisungen sein und das gesamte Konstrukt ist ebenfalls eine Anweisung. (Schachtelung)

```
if (x > y) {if (x > z) {i = x;} else {i = z;}}
        else { if (y > z) {i = y;} else {i = z;}}
```

Welche Klammern können weggelassen werden?

Iteration

Schleife (loop) erlaubt beliebig häufige Wiederholung bestimmter Algorithmusschritte, die im Schleifenkörper stehen.

- **Zählschleife (feste Anzahl von Iterationen). Beispiel:**
  Wiederhole (n-mal) *Anweisung*

Anweisung meint dabei im allgemeinen einen Block (Folge) von Einzelanweisungen. Anweisung bildet den Schleifenkörper und wird bei jedem Durchlauf ausgeführt.

- **Abweisende Schleife:**
  while (Bedingung) *Anweisung*

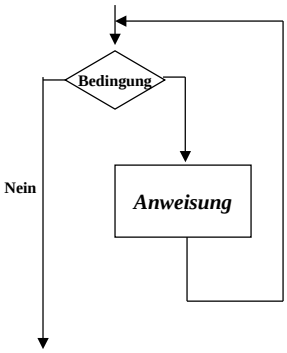
*Die Anweisung wird so oft durchgeführt, wie die Bedingung erfüllt ist. Bedingung wird **vor** Abarbeiten des Schleifenkörpers geprüft.*

- **Akzeptierende Schleife:**
  do *Anweisung* while (Bedingung)

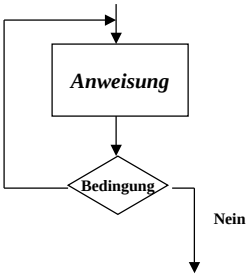
*Die Anweisung wird mindestens einmal ausgeführt. Prüfung **nach** Abarbeitung des Schleifenkörpers.*

Schleifen

Sei B eine Bedingung und *Anweisung* ein Block von Anweisungen, die den Schleifenkörper bilden.



Abweisende Schleife
while (B) *Anweisung*



Akzeptierende Schleife
do *Anweisung* while (B)

Die for Schleife (in C)

Erlaubt übersichtliche Formulierung einfacher Initialisierungs- und Zählvorgänge. Syntax:

for (ausdruck_1; ausdruck_2; ausdruck_3) ***Anweisung***

ist gleichbedeutend mit

```
ausdruck_1;      /*Das ist der Initialisierungsausdruck*/
while (ausdruck_2 ) /*Der zweite Ausdruck legt das Abbruchkriterium fest*/
        {
          Anweisung
         ausdruck_3; /*Wird bei jedem Schleifendurchlauf bewertet. Verändert Wert der Schleifenvariablen*/
        }
```

Beispiel:

```
int i, sum =0, n=100;
for (i=1;i<=n;i++)
        { sum += i; i *=2;}    /*Das ist die Anweisung bzw. der Schleifenkörper*/
```


Rekursion

Definition: Rekursion ist ein allgemeines Verfahren, bei dem ein Konzept direkt oder indirekt durch sich selbst definiert wird.

Beispiel: Eine Liste ist entweder leer oder besteht aus einer Element gefolgt von einer Liste.

Definition: Ein Algorithmus heißt rekursiv, wenn er sich selbst aufruft.

Beispiel: Die Fakultät fac(n) einer ganzen Zahl n > 0 ist definiert durch

$$fac(n) = n * fac(n-1)$$

mit der „Startbedingung“ fac(0) = 1.

Weitere Beispiele: Euklidischer Algorithmus, Türme von Hanoi, Ariadne Faden u.a.

Induktion

Die Induktion ist ein mathematisches Verfahren, um Aussagen über den natürlichen Zahlen zu beweisen.

Einsatz zum Beweis der Richtigkeit rekursiver und iterativer Algorithmen.

Sei S(n) eine beliebige Aussage über eine ganze Zahl n. Der Beweis, dass S(n) für alle Zahlen $n \geq k$ richtig ist geht in folgenden Schritten vonstatten:

- **Induktionsanfang: Beweise die Aussage für ein $k \geq 0$ (meist k=0).**

- **Induktionsschritt: Beweise, dass unter der vorausgesetzten Gültigkeit von S(n) für alle $n \geq k$ auch S(n + 1) richtig ist.**