Bedingte Anweisungen (1)

In Pseudocode:

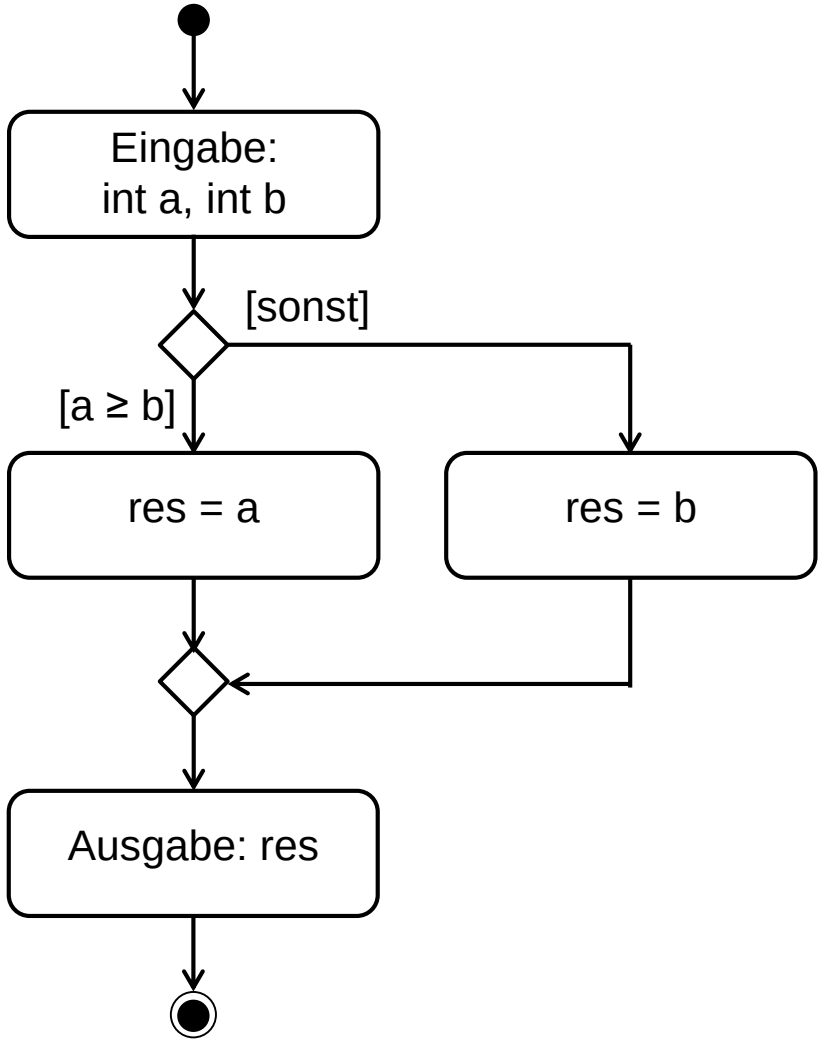
Eingabe: Ganze Zahlen a, b

Ausgabe: Ganze Zahl : Die größere der beiden Zahlen bzw. die Zahl, falls beide gleich sind

Berechnung:

- Falls a ≥ b:
  Setze res auf a
  Sonst:
  Setze res auf b
- Die Ausgabe ist res

Als Diagramm:



Bedingte Anweisungen (2)

- Der zweite Zweig (Sonst:) kann auch leer sein.

In Pseudocode:

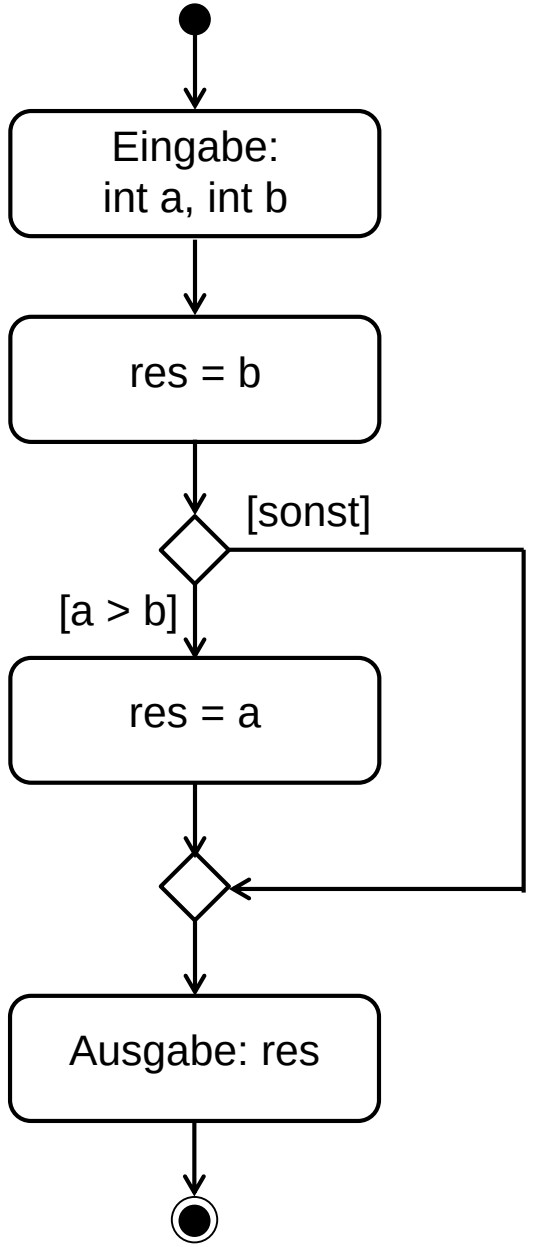
Eingabe: Ganze Zahlen a, b

Ausgabe: Ganze Zahl : Die größere der beiden Zahlen bzw. die Zahl, falls beide gleich sind

Berechnung:

- Setze res auf b
- Falls a > b:
  Setze res auf a
- Die Ausgabe ist res

Als Diagramm:



Bedingte Anweisungen (3)

In Pseudocode:

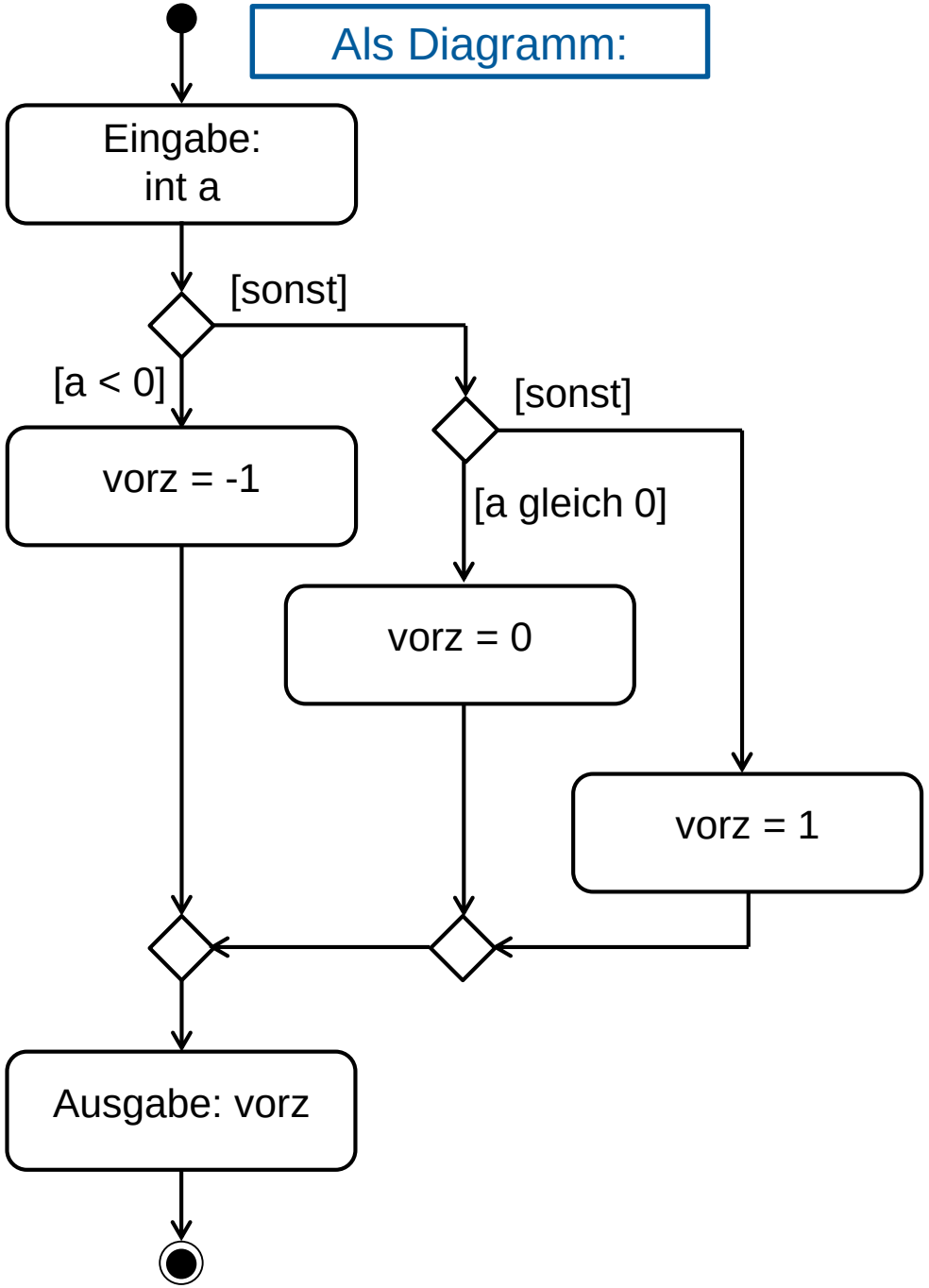
Eingabe: Ganze Zahl a

Ausgabe: Ganze Zahl : -1, 0, 1 je nachdem, ob a negativ, gleich Null oder positiv ist

Berechnung:

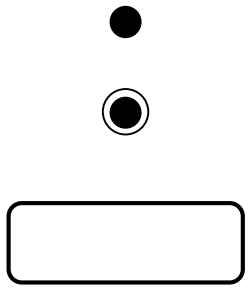
- Falls a < 0:
  Setze vorz auf -1
  Sonst:
  Falls a gleich 0:
  Setze vorz auf 0
  Sonst:
  Setze vorz auf 1
- Die Ausgabe ist vorz

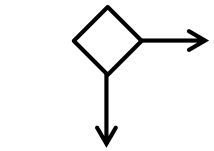
Als Diagramm:

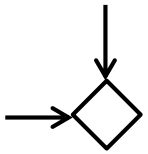


Bedingte Anweisungen (4)

- **Ablaufdiagramme** haben…
  - genau einen Eingang (dargestellt durch einen ausgefüllten Kreis),
  - beliebig viele Ausgänge (dargestellt durch Doppelkreise),
  - beliebig viele Anweisungen (in Vierecken mit abgerundeten Ecken),
  - beliebig viele binäre Verzweigungen, d.h. Verzweigungen mit zwei ausgehenden Pfeilen (dargestellt durch Rauten), wobei
    - an beiden ausgehenden Pfeilen eine Bedingung in eckigen Klammern steht,
    - jede Bedingung die logische Verneinung der anderen Bedingung oder das Wort „sonst“ ist,
  - beliebig viele binäre Zusammenführungen von Zweigen (ebenfalls dargestellt durch Rauten),
  - Pfeile zur Darstellung des Ablaufs des Programms.







Bedingte Anweisungen (5)

- In den Bedingungen von Schleifen und bedingten Anweisungen werden oft Vergleiche verwendet, z.B. x < 1 – y. Der Vergleich ist wahr z.B. für x = –1, y = 0 und ist falsch z.B. für x = 2, y = –3
- Folgende **Vergleichsoperatoren** können auf Zahlen und auf Zeichenketten angewendet werden:

Operator	Prüfung auf…
<	kleiner
<=	kleiner gleich
>	größer
>=	größer gleich
==	Gleichheit
!=	Ungleichheit

- Für Zahlen ist die Bedeutung der Operatoren wie in der Mathematik.

Bedingte Anweisungen (6)

- Zeichenketten werden Zeichen für Zeichen verglichen, z.B.
  "bella" < "besla", weil "b" == "b", "e" == "e", "l" < "s"

b	e	l	l	a
‖	‖	^		
b	e	s	l	a

- Einzelne Zeichen werden nach ihrer Codierung verglichen. Es ist zu beachten, dass Großbuchstaben kleiner als Kleinbuchstaben sind, z.B. "Bella" < "bella".
- Weitere Beispiele:
  "bella" < "donna", "bella" < "belladonna", "bel a" < "bella"

Bedingte Anweisungen (7)

- Bedingungen können mit **logischen Operatoren** zusammengesetzt werden:

Operator	Bedeutung
and	logisches Und
or	logisches Oder
not	logisches Nicht

- Beispiel:

```
x > 0 and y > 0
```

- Die logischen Operatoren werden durch folgende Tabellen definiert:

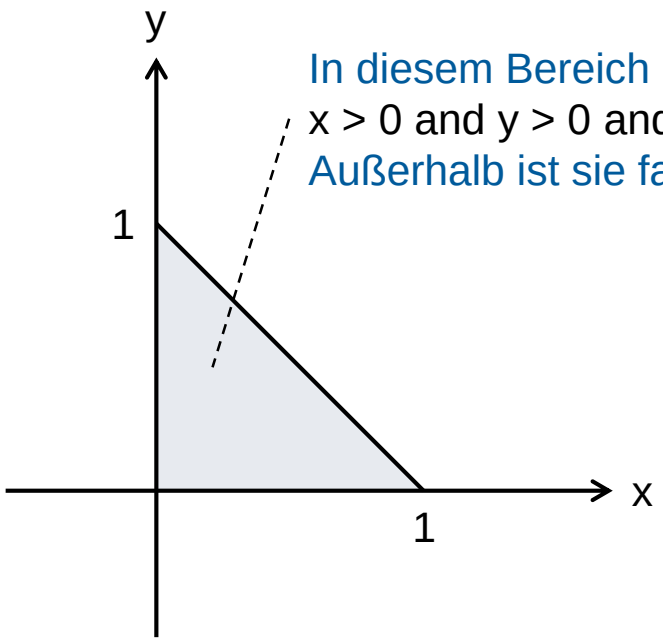
x	y	x and y
wahr	wahr	wahr
wahr	falsch	falsch
falsch	wahr	falsch
falsch	falsch	falsch

x	y	x or y
wahr	wahr	wahr
wahr	falsch	wahr
falsch	wahr	wahr
falsch	falsch	falsch

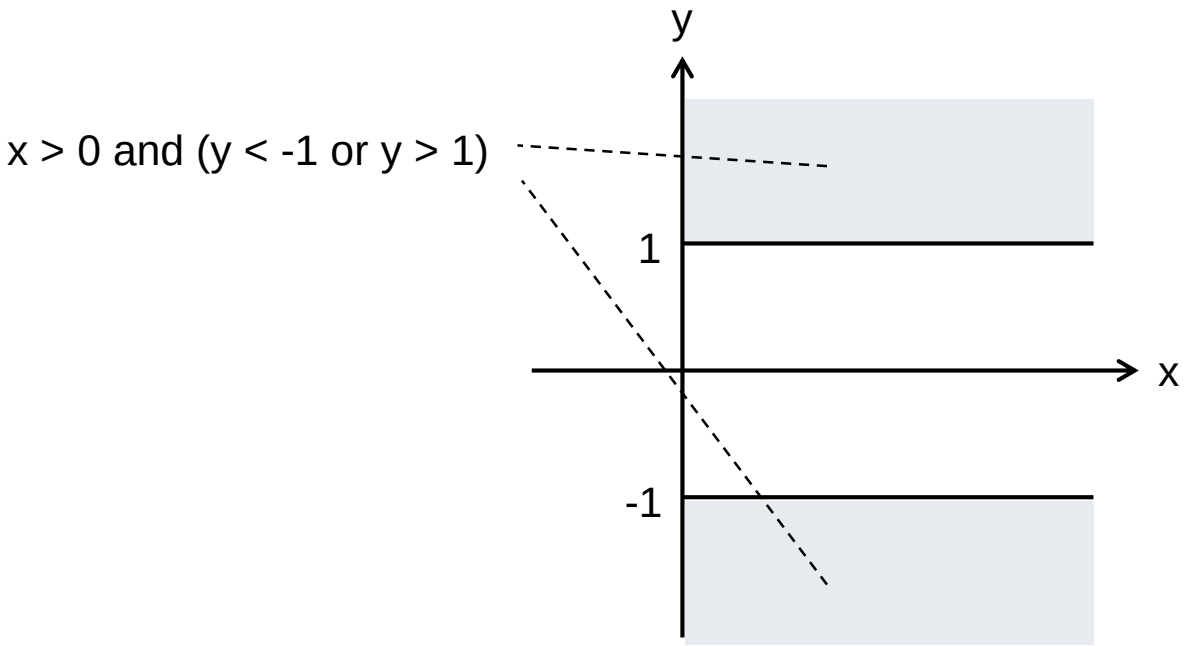
x	not x
wahr	falsch
falsch	wahr

Bedingte Anweisungen (8)

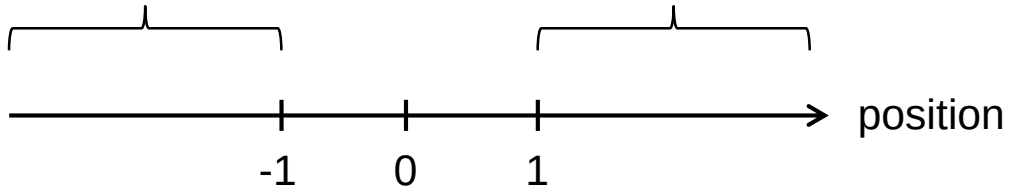
- Beispiele:



In diesem Bereich ist die Bedingung
x > 0 and y > 0 and y < 1 – x wahr.
Außerhalb ist sie falsch.



In diesen Bereichen ist
position < -1 or position > 1 wahr.



Die Bedingung position < -1 or position > 1 ist gleichbedeutend mit
not (position >= -1 and position <= 1)

Bedingte Anweisungen (9)

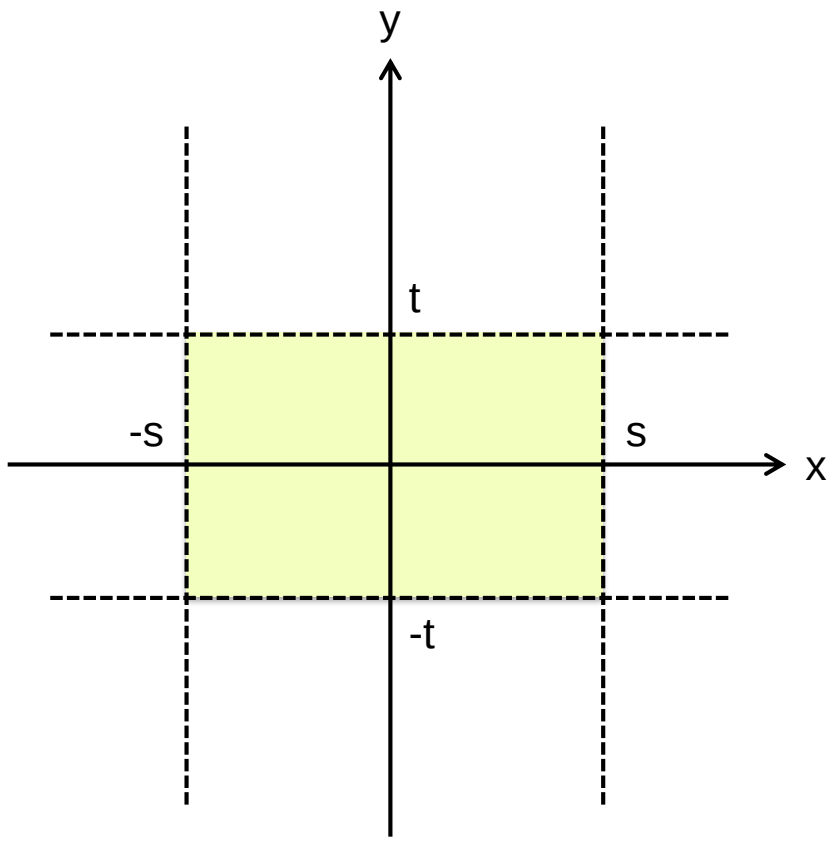
Frage

- Welche Punkte (x1, y1) und (x2, y2) erfüllen die Bedingung

  (abs(x1) > s or abs(y1) > t) and (abs(x2) > s or abs(y2) > t)

  falls s und t positive Zahlen sind? abs(z) ist der Absolutbetrag einer Zahl z.

	abs(x1) > s or abs(y1) > t	abs(x2) > s or abs(y2) > t



Bedingte Anweisungen (10)

Aufgaben

Erstellen Sie Ablaufdiagramme für folgende Programme.

1. Eingabe: Ganze Zahlen a, b mit a > 0 und b > 0
   Ausgabe: Ganze Zahl : a modulo b (Rest bei ganzzahliger Division)

   Es soll solange b von a subtrahiert werden, bis a kleiner als b ist. Die Bedingungen a > 0 und b > 0 müssen nicht überprüft werden.

2. Eingabe: Ganze Zahlen a, b mit a > 0 und b > 0
   Ausgabe: Ganze Zahl : Der größte gemeinsame Teiler von a und b

   Es soll solange die kleinere der beiden Zahlen von der größeren subtrahiert werden, bis beide Zahlen gleich sind.

   Was passiert, wenn eine Zahl < 0 eingegeben wird? Ergänzen Sie den Algorithmus durch if-Abfragen, die feststellen, ob eine der beiden Eingaben ≤ 0 sind. In diesem Fall soll die Zahl –1 ausgegeben werden.

Bedingte Anweisungen (11)

Aufgaben

3. Eingabe: Zahlen a, b, c
   Ausgabe: Zahl : Die größte der drei Zahlen

4. Eingabe: Zahlen z_1, z_2, …, z_n
   Ausgabe: Die Zahlen in sortierter Reihenfolge, kleinste Zahl zuerst

5. „Zu einer gegebenen Zeit in Stunden, Minuten und Sekunden sollen Sie eine Sekunde hinzuzählen. Aus der Uhrzeit 12:59:59 wird beispielsweise nach der Addition von einer Sekunde die Zeit 13:00:00 Uhr. Lesen Sie Stunden-, Minuten- und Sekundenwerte der Uhrzeit in drei Integer-Werte ein.“ Die Ausgabe ist die neue Uhrzeit. (Klein 2018, Seite 74)

Bedingte Anweisungen (12)

Aufgaben

6. „In der nächsten Aufgabe lernen wir einen besonderen Frosch kennen, so wie ihn sich nur Mathematiker ausdenken können. Besonders seine Art, eine Straße zu überqueren, macht es zweifelhaft, ob er in der realen Welt lange überleben könnte. Er überquert eine 2,50 Meter breite Straße wie folgt: Mit dem ersten Sprung legt er die erstaunliche Distanz von einem Meter zurück, dann springt er wegen zunehmender Erschöpfung mit jedem weiteren Schritt immer nur noch halb so weit wie vorher. Die Entfernung, die er dabei zurücklegt, berechnet sich also als Summe der Werte 1+0,5+0.25+0,125 und so weiter. […] Versuchen Sie […] herauszubekommen, ob der Frosch es auf die andere Straßenseite schafft.“ (Klein 2018, Seite 84-85)

Algorithmen (1)

- Eine Verarbeitungsvorschrift mit den folgenden Eigenschaften heißt **Algorithmus**:
  - Eingabespezifikation: „Es muss genau spezifiziert sein, welche Eingabegrößen erforderlich sind und welchen Anforderungen diese Größen genügen müssen, damit das Verfahren funktioniert.“
  - Ausgabespezifikation: „Es muss genau spezifiziert sein, welche Ausgabegrößen (Resultate) mit welchen Eigenschaften berechnet werden.“
  - Endliche Beschreibung: „Das Verfahren muss in einem endlichen Text vollständig beschrieben sein.“
  - Effektivität: „Jeder Schritt des Verfahrens muss effektiv (d.h. tatsächlich) mechanisch ausführbar sein.“
  - Determiniertheit: „Der Verfahrensablauf ist zu jedem Zeitpunkt fest vorgeschrieben.“
  - Partielle Korrektheit: „Jedes berechnete Ergebnis genügt der Ausgabespezifikation, sofern die Eingaben der Eingabespezifikation genügt haben.“
  - Terminierung: „Der Algorithmus hält nach endlich vielen Schritten mit einem Ergebnis an, sofern die Eingaben der Eingabespezifikation genügt haben.“

Aus: W. Küchlin und A. Weber (2005) *Einführung in die Informatik: Objektorientiert mit Java*. Springer, 3. Aufl.

Algorithmen (2)

- Die Eigenschaften „Endliche Beschreibung“, „Effektivität“ und „Determiniertheit“ sind bei der Formulierung als Programm automatisch erfüllt.
- Die Aufgabe des Programmierers ist es vor allem, die partielle Korrektheit und die Terminierung sicherzustellen.
  - Partielle Korrektheit: Wird vor allem durch Testen geprüft. Verwenden Sie in den Aufgaben und im Praktikum genügend viele Testfälle, um sich von der partiellen Korrektheit zu überzeugen. Als Testfälle können Sie einfach Testaufrufe des Algorithmus bzw. der Funktion im Hauptprogramm schreiben.
  - Terminierung: Damit ist die Terminierung von Schleifen, d.h. dass die Schleifenbedingung irgendwann falsch wird, und die Terminierung von rekursiven Funktionen (später mehr dazu) gemeint. Auch dafür sind Testfälle nützlich.

Der Begriff Algorithmus ist hergeleitet vom Namen des persischen Mathematikers Muhammed Ibn-Musa Al-Chwarizmi (780 – 830?).

https://commons.wikimedia.org/wiki/File:1983_CPA_5426_(1).png