Algorithmen

Begriffsbestimmung

- Ein **Algorithmus** ist eine eindeutige, endliche Beschreibung eines allgemeinen, endlichen Verfahrens zur schrittweisen Ermittlung gesuchter Größen aus gegebenen Größen.

Ein einfaches Beispiel für einen nicht eindeutigen Algorithmus ist das Raten einer Zahl zwischen 1 und 10:

1. Gegeben sei eine geheime Zahl zwischen 1 und 10.
2. Rate eine Zahl zwischen 1 und 10.
3. Vergleiche die geratene Zahl mit der geheimen Zahl.
4. Wenn die geratene Zahl gleich der geheimen Zahl ist, wurde die Zahl erfolgreich erraten.
5. Falls die geratene Zahl nicht gleich der geheimen Zahl ist, wiederhole Schritt 2 bis 4.

Dieser Algorithmus ist nicht eindeutig, da das Raten einer Zahl auf verschiedene Weise geschehen kann: zufällig, sequenziell (1, 2, 3, ...), oder auf andere Weise. Es gibt keinen festen Weg, um die Zahl beim ersten Versuch zu erraten, und die Anzahl der benötigten Schritte variiert je nach gewählter Methode und der geheimen Zahl.

Algorithmen

Begriffsbestimmung

- Ein **Algorithmus** ist eine eindeutige, endliche Beschreibung eines allgemeinen, endlichen Verfahrens zur schrittweisen Ermittlung gesuchter Größen aus gegebenen Größen.

Ein einfaches Beispiel für einen nicht endlichen Algorithmus ist ein Algorithmus, der versucht, die Zahlen von 1 bis unendlich aufzulisten:

1. Setze die aktuelle Zahl auf 1.
2. Gib die aktuelle Zahl aus.
3. Erhöhe die aktuelle Zahl um 1.
4. Wiederhole Schritte 2 und 3.

Dieser Algorithmus ist nicht endlich, weil er für immer weitergeht und keine abgeschlossene Liste von Zahlen erzeugt. Da er kontinuierlich läuft, gibt es keine festgelegte Anzahl von Schritten, die zur Erstellung einer endlichen Liste führen würden.

Algorithmen

Begriffsbestimmung

- Ein **Algorithmus** ist eine eindeutige, endliche Beschreibung eines allgemeinen, endlichen Verfahrens zur schrittweisen Ermittlung gesuchter Größen aus gegebenen Größen.

Ein einfaches Beispiel für einen Algorithmus, der nicht allgemein ist, wäre ein Algorithmus, der lediglich die Summe von 2 und 3 berechnet:

1. Gegeben seien die Zahlen 2 und 3.
2. Addiere die beiden Zahlen.
3. Gib das Ergebnis der Addition aus.

Dieser Algorithmus ist nicht allgemein, da er nur für diese spezifische Kombination von Zahlen funktioniert (2 und 3) und nicht für andere Zahlen oder Fälle erweitert werden kann. Ein allgemeinerer Algorithmus wäre zum Beispiel ein Algorithmus zur Addition von zwei beliebigen Zahlen.

Algorithmen

Begriffsbestimmung

- Ein **Algorithmus** ist eine eindeutige, endliche Beschreibung eines allgemeinen, endlichen Verfahrens zur schrittweisen Ermittlung gesuchter Größen aus gegebenen Größen.

- Ausführbarkeit:
  - Die Anweisungen eines Algorithmus müssen ausführbar sein, d.h. eine Person, die den Formalismus kennt und die elementaren Algorithmen beherrscht, muss ihn in ein äquivalentes, ausführbares Programm überführen können.

- Gegenbeispiel:
  - Eine Vorschrift wie „Wenn man in 14 Tagen Zahnschmerzen bekommt, dann vereinbart man bereits heute einen Termin mit dem Zahnarzt, um die Wartezeit zu verkürzen.“ ist in der Realität nicht ausführbar.

Algorithmus vs. Programm

- Algorithmus:
  1. `einlesen b`
  2. `einlesen x`
  3. `erg ← 4 * Logarithmus(b,x)`
  4. `ausgeben erg`

- Programm:

```
public class VierfacherLogarithmus
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);

    int b = sc.nextInt();
    int x = sc.nextInt();
    int erg = 4 * berechneLogarithmus(b, x);
    System.out.println(erg);
  }
}
```

- Im Programm wird der Algorithmus durch die elementaren Bausteine der Programmiersprache beschrieben.
  - Die Programmiersprache gibt damit vor, welche elementaren Algorithmen verwendet werden können.

- Verschiedene Programmiersprachen haben:
  - Verschiedene elementare Bausteine
  - Somit unterschiedliche Programme

- Dennoch: derselbe Algorithmus!

- Folgerung: Algorithmen sollten unabhängig von einer Programmiersprache beschrieben werden.

Algorithmen

Algorithmus vs. Programm

- Formalisierungsmöglichkeiten für Algorithmen:
  - Freitext: „Zuerst wird die Variable n auf 0 gesetzt. Dann…“
  - Struktogramme
  - Flussdiagramme
  - Pseudocode

Pseudocode ist eine informelle Art, Algorithmen zu beschreiben, ohne die strenge Struktur und Syntax einer Programmiersprache zu verwenden. Es ist ein nützliches Werkzeug, um den Ablauf eines Algorithmus zu skizzieren und zu verstehen, bevor man ihn in Code umsetzt. Hier ist ein einfaches Tutorial für Pseudocode mit Beispielen:

1. Verwendung von natürlicher Sprache: Pseudocode verwendet natürliche Sprache und einfache Programmierstrukturen, um die Logik eines Algorithmus darzustellen. Es ist nicht auf eine bestimmte Programmiersprache beschränkt und kann von Programmierern mit unterschiedlichen Hintergründen verstanden werden.
2. Grundlegende Strukturen: Pseudocode verwendet grundlegende Programmierstrukturen wie Sequenzen, Schleifen und Entscheidungen. Hier sind einige Beispiele:

- Sequenz: Eine Sequenz von Anweisungen, die nacheinander ausgeführt werden.

- `if`-Anweisungen benutzen ebenfalls in Pascal-Notation:

```
if <Bedingung> then
  <Anweisungen>
else
  <Anweisungen>
endif
```

- Schleifen benutzen ebenfalls die in Pascal übliche Syntax:

```
while <Bedingung> do
  <Anweisungen>
end while

repeat
  <Anweisungen>
until <Bedingung>

for <Zähler> ← <Start> to <Ende> do
  <Anweisungen>
end for
```

  - Varianten für `for`-Schleifen:

```
for <Zähler> ← <Start> to <Ende> step <Schrittweite> do
for <Zähler> ← <Start> downto <Ende> do
for <Zähler> ← <Start> downto <Ende> step <Schrittweite> do
```

Pseudocode

Arrays, Ein- und Ausgabe

- Arrays beginnen **im Gegensatz zu Java** grundsätzlich mit dem Index 1:

```
A[1] ← 5        // Erstes Element bekommt 5 zugewiesen
A[0] ← 5        // Fehlerhafte Anweisung
```

- Bei der Ein- und Ausgabe wird vom technischen Vorgang abstrahiert:

```
einlesen a
a ← a+1
ausgeben a
```

- Weitere Informationen und Beispiele finden Sie in der Datei `Pseudocode.pdf` im ILIAS-System.
  - Bis nächste Woche lesen, verstehen und anwenden können!

Pseudocode

Beispiel

- **Hauptalgorithmus:**

```
einlesen b                  // Abstraktion vom techn. Vorgang
einlesen x
erg ← 4 * Logarithmus(b,x)
ausgeben erg                // Abstraktion vom techn. Vorgang
```

- **Funktion:**

```
Logarithmus(b,x)
  n ← 0                     // Zuweisung
  y ← x                     // Zuweisung
  while y >= b do           // Schleife
    y ← y/b
    n ← n+1
  end while
  return n
```

Beispiele:

- Beispiel 1: Berechnung der Summe der ersten n natürlichen Zahlen.

```
FUNCTION SummeErsterNZahlen(n)
  SET summe = 0
  FOR i = 1 TO n
    summe = summe + i
  END FOR
  RETURN summe
END FUNCTION
```

- Beispiel 2: Lineare Suche in einem Array.

```
FUNCTION LineareSuche(array, gesuchteZahl)
  FOR i = 0 TO Länge(array) - 1
    IF array[i] == gesuchteZahl
      RETURN i
    END IF
  END FOR
  RETURN -1
END FUNCTION
```

Aufgaben:

Das „Sieb des Eratosthenes“ ist ein von dem griechischen Philosophen Eratosthenes (276-195 v. Chr.) entwickelter Algorithmus zur Berechnung aller Primzahlen bis zu einer vorgegebenen natürlichen Zahl n. Der Algorithmus in Umgangssprache (angelehnt an den „Duden der Informatik“):

1. *Man schreibe alle Zahlen von 1 bis n hin und streiche die Zahl 1 durch.*
2. *Sei i die kleinste noch nicht durchgestrichene und nicht eingerahmte Zahl. Solange i existiert und $i^2 \leq n$ ist, rahme man i ein und streiche alle Vielfachen von i durch (die Vielfachen von i werden „ausgesiebt“).*
3. *Die eingerahmten und die nicht durchgestrichenen Zahlen sind die Primzahlen von 1 bis n.*

a) Führen Sie den Algorithmus zunächst auf einem Blatt Papier für die Zahlen von 1 bis 25 durch:

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**
✘	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Euklidische Algorithmus zur Berechnung des größten gemeinsamen Teilers (ggT) zweier Zahlen. Dieser Algorithmus wurde von dem griechischen Mathematiker Euklid in seinem Werk "Die Elemente" beschrieben.

1. Gegeben seien zwei natürliche Zahlen a und b (a ≥ b).
2. Falls b = 0, ist der ggT gleich a. Andernfalls fahre fort mit Schritt 3.
3. Dividiere a durch b und berechne den Rest r.
4. Setze a = b und b = r.
5. Wiederhole Schritte 2 bis 4, bis b = 0.

Aufgabe: Schreiben Sie die Java-Klasse **`EuklidischerAlgorithmus`**, die den obigen Algorithmus realisiert und den ggT von zwei vom Benutzer eingegebenen Zahlen berechnet und ausgibt.

Sortieren von Zahlen in einem Array mit dem Bubble-Sort-Algorithmus. Bubble Sort ist ein einfacher Sortieralgorithmus, der auf dem Prinzip des wiederholten Vergleichs benachbarter Elemente basiert und diese vertauscht, wenn sie in der falschen Reihenfolge sind.

1. Gegeben sei ein Array mit Zahlen.
2. Beginne am ersten Element des Arrays und vergleiche es mit dem benachbarten Element.
3. Falls das erste Element größer ist als das benachbarte Element, vertausche sie.
4. Gehe zum nächsten Element und wiederhole Schritte 2 und 3, bis das Ende des Arrays erreicht ist.
5. Wiederhole Schritte 2 bis 4 für das gesamte Array, bis keine Vertauschungen mehr notwendig sind.

Aufgabe: Schreiben Sie die Java-Klasse **`BubbleSort`**, die den obigen Algorithmus realisiert und ein Array von Zahlen sortiert und ausgibt.