Ein **Algorithmus** in der Informatik bzw. Mathematik ist ein **Verfahren**, mit dem man eine Berechnung durchführen oder eine bestimmte Aufgaben lösen kann. Der Begriff geht auf den choresmischen Gelehrten al-Chwarizmi (geb. 780 n. Chr.) zurück.

Typische Problemstellungen sind zum Beispiel:

- Berechne die Quadratwurzel einer Zahl mithilfe der Grundrechenarten (z.B. mit dem „Heron-Verfahren“)
- Bestimme den kürzesten Wegs zwischen zwei beliebigen Orten
- Sortiere eine Liste von Adressen nach den Nachnamen

Es gibt verschiedene Möglichkeiten, Algorithmen darzustellen:

- **umgangssprachlich**
- als UML-**Aktivitätsdiagramm**
- als **Quellcode** in einer Programmiersprache

Beispiel

Für die folgende Aufgabe soll ein **Algorithmus** formuliert werden:
Ein Array soll linear mit Werten gefüllt werden, angefangen bei –75, in 25er-Schritten.

Umgangssprache

Setze eine Variable „a“ auf -75.

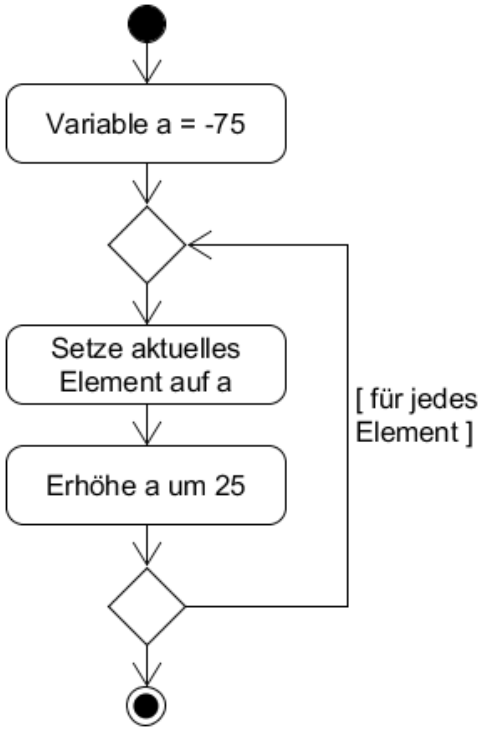
Wiederhole für alle Elemente des Arrays:

- Setze das aktuelle Element auf den Wert der Variablen
- Erhöhe die Variable um 15.

Programmcode

```
public void fuelleLinear()
{
   int i, a;
   a = -75;
   for (i=0; i<liste.length; i++)
   {
      liste[i] = a;
      a = a + 25;
   }
}
```

Aktivitätsdiagramm



Aufgabe

Die folgenden Algorithmen sind als Java-Quellcode dargestellt.
Um den Quellcode zu verstehen, fülle jeweils die Tabelle mit den Werten des Arrays aus.
Stelle den Algorithmus dann jeweils 1. in Umgangssprache, 2. als Aktivitätsdiagramm dar.

a) Lineare Füllung in 20er-Schritten

```
public void linear20()
{
   int i;
   for (i = 0; i < liste.length; i++)
   {
      liste[i] = 20 * i;
   }
}
```

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

b) Füllung mit den „Dreieckszahlen“

```
public void dreieck()
{
   int i, wert;
   wert = 0;
   for (i = 0; i < liste.length; i++)
   {
      wert = wert + i;
      liste[i] = wert;
   }
}
```

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

c) Füllen mit der „Fibonacci-Folge“

```
public void fibonacci()
{
   int i;
   liste[0] = 1;
   liste[1] = 1;
   for (i = 2; i < 10; i++)
   {
      liste[i] = liste[i-1] + liste[i-2];
   }
}
```

0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*