Aufgaben

1 Fibonacci-Spiralen

a) Implementieren Sie innerhalb einer Klasse FIBONACCI die rekursiv definierte Funktion *fib* des Lehrtextes und berechnen Sie *fib(8)*.

b) Die Nautilus-Schnecke hat im Laufe ihres Lebens ein Schneckenhaus gebildet, dessen Fibonacci-Spirale aus drei Vollumdrehungen besteht. Implementieren Sie eine Funktion *spirallängeGeben*, welche die Gesamtlänge in Abhängigkeit der Anzahl der Vollumdrehungen berechnet. Beachten Sie, dass sich die Spirale aus Viertelkreisen zusammensetzt.

2 Gitterpunkte

Ein Paar aus jeweils n zueinander parallelen Geraden schneidet sich (Fig. 1). Wie viele Schnittpunkte gibt es für $n = 37$, wenn für $n = 36$ die Anzahl A der Schnittpunkte $A(36) = 1296$ beträgt? Wie könnte man allgemein die Anzahl der Schnittpunkte von n Geradenpaaren aus der Anzahl für $n-1$ bestimmen? Bestimmen Sie eine rekursive Definition der Funktion A(n). Geben Sie ebenso die Abbruchbedingung an.

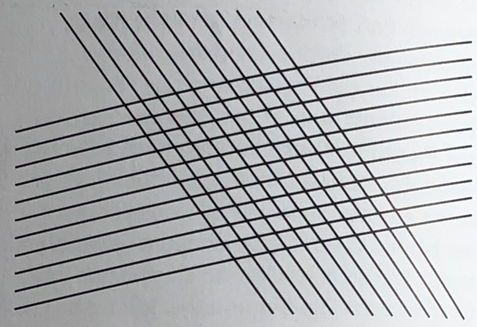


Fig. 1

3 Bakterienwachstum

Eine Bakterienkultur vermehrt sich jede Stunde um 20 %. Zu Beginn der Beobachtung besteht die Kultur aus 300 Bakterien.

a) Geben Sie eine Rekursionsvorschrift für die Bestimmung der Bakterienanzahl nach n Stunden an.

b) Implementieren Sie die Funktion und berechnen Sie, wie viele Bakterien nach 24 Stunden vorhanden sind.

4 Zeichenketten II

a) Beschreiben Sie die Funktionalität der Funktion *funktion1*. Ermitteln Sie beispielsweise die Funktionswerte für die Argumente „HANNAH“, „SARAH“ und „OTTO“.

Hinweis: Informieren Sie sich über die verfügbaren Methoden der Klasse `String` in der *Java*-API.

Java

```
public boolean funktion1(String s) {
  if ((s.length() == 0) || (s.length() == 1)) {
    return true;
  }
  else {
    return funktion1(s.substring(1, s.length() - 1))
        && (s.charAt(0) == s.charAt(s.length() - 1));
  }
}
```

b) Implementieren Sie eine weitere rekursiv definierte Funktion *umdrehen*, welche Wörter umdreht, d.h. zum Beispiel, dass das Argument „REGAL“ das Wort „LAGER“ liefert.

c) Wie könnte man mithilfe der Funktion *umdrehen* die in Teilaufgabe a) dargestellte Funktionsdefinition vereinfachen?

5 Dreiecksmuster

Fig. 1 zeigt das dynamische Datenflussdiagramm einer rekursiv definierten Funktion *anzahl*, welche für die Eingabe 3 die Anzahl aller enthaltenen und umbeschriebenen Dreiecke einer Dreiecksfigur berechnet (Fig. 2).

a) Bestimmen Sie die Funktionsdeklaration mit Rekursionsgleichung und Abbruchbedingung.

b) Geben Sie für den Fall $n = 3$ die Termdarstellung der Funktionsaufrufe für die schrittweise Berechnung an.

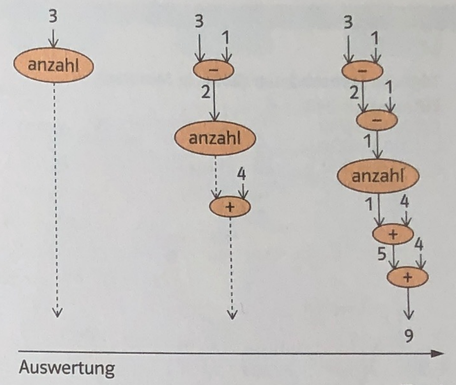
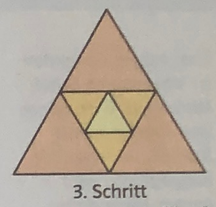


Fig. 1



3. Schritt

Fig. 2

6 Zahnradgetriebe

a) Für den größten gemeinsamen Teiler $ggT(a, b)$ und das kleinste gemeinsame Vielfache $kgV(a, b)$ zweier natürlicher Zahlen a und b gilt der Zusammenhang $ggT(a, b) \cdot kgV(a, b) = a \cdot b$.

Weisen Sie damit nach, dass die folgende rekursive Funktionsdeklaration für *kgV* gilt:

$$kgV(a, b) = \begin{cases} kgV(a-b, b) \cdot \frac{a}{a-b}, & falls\ a > b \\ kgV(a, b-a) \cdot \frac{b}{b-a}, & falls\ a < b \\ a, & falls\ a = b \end{cases}.$$

b) Zwei Zahnräder mit verschiedener Anzahl von Zähnen greifen ineinander. Für die Anzahl der notwendigen Umdrehungen, sodass dieselben Zähne wieder zusammenkommen, benötigt man das *kgV* aus der Anzahl der Zähne der beiden Zahnräder.

Implementieren Sie in einer Klasse ZAHNRADGETRIEBE die rekursiv definierte Funktion *kgV* und eine weitere Funktion *anzahlUmdrehungen*, welche die Anzahl der Umdrehungen der beiden Zahnräder als Text ausgibt.

c) Zeichnen Sie ein dynamisches Datenflussdiagramm zur Auswertung von $kgV(8, 12)$.

7 Das Märchen vom Reiskorn

Der Legende nach stammt das Schachspiel aus Indien und begeisterte den König damals so sehr, dass er seine Armee nach dem Erfinder des Spiels suchen ließ. Aus Dankbarkeit wollte er seinem Untertan (angeblich ein Mathematiklehrer namens Buddhiram) jeden Wunsch erfüllen. Dieser wünschte sich daraufhin ein Reiskorn auf dem ersten Feld des Schachbretts, zwei auf dem zweiten, vier auf dem dritten usw., also jeweils die doppelte Anzahl wie auf dem vorhergehenden Feld. Der König wollte dem Untertan seinen Wunsch erfüllen und musste bald feststellen, dass im ganzen Reich nicht genügend Reiskörner vorhanden waren, sodass der Untertan zum reichsten Mann im Land wurde.

a) Zur Berechnung der Anzahl der Reiskörner auf dem letzten Feld hat Matthias folgende Methode geschrieben:

```
public long potenzVonZwei(int n) {
  return 2 * potenzVonZwei(n - 1);
}
```

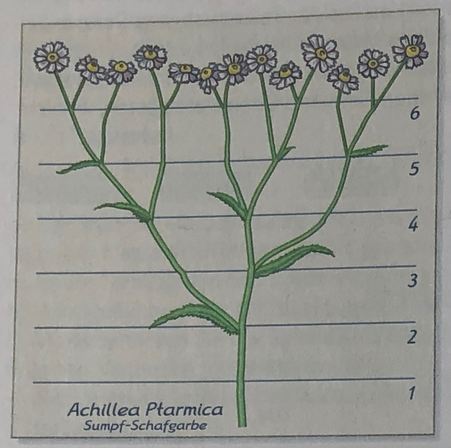
Warum funktioniert die Methode nicht? Was hat Matthias vergessen? Korrigieren Sie seinen Fehler.

b) Mit welchem Aufruf lässt sich dann die Anzahl der Reiskörner auf dem 64. Feld des Schachbretts berechnen? Warum liefert *Java* ein unsinniges Ergebnis?

c) Schreiben Sie eine weitere Methode, die die Gesamtsumme der Reiskörner bis zum jeweiligen Feld berechnet.

Hinweis zu c):
Ihre Methode kann die Methode `potenzVonZwei(n)` *aufrufen.*

3 Rekursive Funktionen



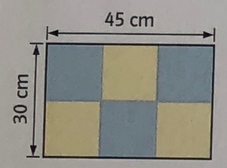
Bei der Sumpf-Schafgarbe wächst ein neuer Trieb zwei Monate, bevor er einen neuen Seitentrieb bilden kann. Der alte Trieb bringt hingegen im weiteren Wachstumsverlauf pro Monat einen neuen Trieb hervor. Die Anzahl A(n) der Trieb-Enden am Ende des n-ten Monats lässt sich aus der Anzahl der Trieb-Enden am Ende der beiden vorangegangenen Monate berechnen.
Wie viele Trieb-Enden (Blüten) hat die Sumpf-Schafgarbe am Ende des siebten Monats? Wie sieht der allgemeine Term für A(n) aus?

Manche Funktionen enthalten in ihrem Funktionsterm (eventuell sogar an mehreren Stellen) den eigenen Funktionsbezeichner. Solche Funktionen heißen **rekursiv**.

Deklaration rekursiver Funktionen

Die Berechnung des größten gemeinsamen Teilers (*ggT*) zweier Zahlen lässt sich auf die Berechnung des *ggT* der kleineren der beiden Zahlen und ihrer Differenz zurückführen. Dieser Schritt wird so lange wiederholt, bis die Argumente gleich groß sind. Dieser Wert entspricht dann dem gesuchten *ggT*:

Um ein Rechteck mit 45 cm Länge und 30 cm Breite mit möglichst großen Quadraten auszulegen, muss die Seitenlänge des Quadrates ggT(45,30) cm betragen.



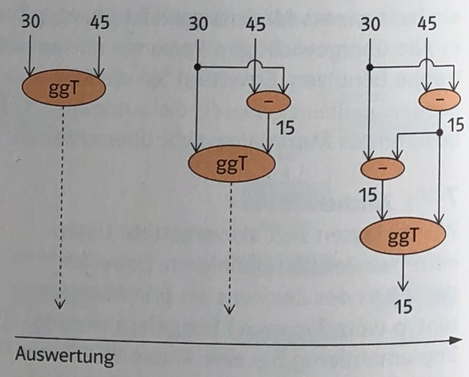
Rekursionsvorschrift:

$$ggT(a, b) = \begin{cases} ggT(a-b, b), & \text{falls } a > b \\ ggT(a, b-a), & \text{falls } a < b \\ a, & \text{falls } a = b \text{ (Abbruchbedingung)} \end{cases}$$

$(a, b \in \mathbb{N},\ a > 0,\ b > 0)$

Dynamische Datenflussdiagramme

Die Berechnung einer rekursiv definierten Funktion endet, sobald die Abbruchbedingung erfüllt ist. In allen anderen Fällen wird ein neuer Prozess derselben Funktion erzeugt. Zur Darstellung der Funktionsauswertung reicht daher ein einziges Datenflussdiagramm nicht aus: man benötigt eine Sequenz von Datenflussdiagrammen. Die gestrichelten Datenflusspfeile symbolisieren die Ausgänge, die noch nicht mit einem Wert belegt werden können.



Bei der Funktion ggT handelt es sich genauer um eine linear rekursive Funktion, da jede Funktionsauswertung zu höchstens einem weiteren rekursiven Aufruf führt.

Die Folge der einzelnen Schritte einer Funktionsauswertung bezeichnet man als **Aufrufsequenz**, z. B. für die Auswertung von *ggT(30, 45)*:
ggT(30, 45) = ggT(30, 45 − 30) = ggT(30 − 15, 15) = 15.

Bei der Implementierung der rekursiv definierten Funktion *ggT* verwendet man für die Prüfung der Abbruchbedingung bzw. der verschiedenen Fälle der Rekursionsgleichung geschachtelte Alternativen (Fig. 1).

Java

```
public int ggT(int a, int b) {
  // Abbruchbedingung
  if (a == b) {return a;}
  // Fälle mit Rekursionstermen
  else if (a > b) {return ggT(a - b, b);}
  else {return ggT(a, b - a);}
}
```

Fig. 1

Kaskadenartige Rekursion

In vielen Gemüsearten oder auch in den Schalen von Schnecken findet sich die Struktur der Fibonacci-Spirale wieder (Fig. 2): Für die Fortsetzung der Konstruktion zeichnet man an die längere Seite des aus Quadraten bestehenden Rechtecks ein weiteres Quadrat, dessen Seitenlänge sich damit aus der Summe der Seitenlängen der beiden zuletzt gezeichneten Quadrate ergibt. Der entsprechende Viertelkreis im Quadrat setzt die begonnene Spirale fort.

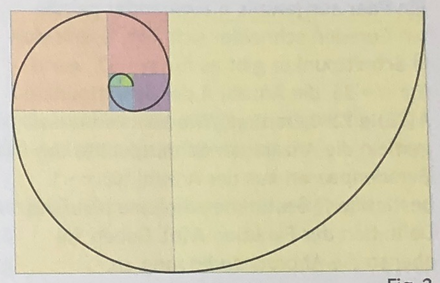


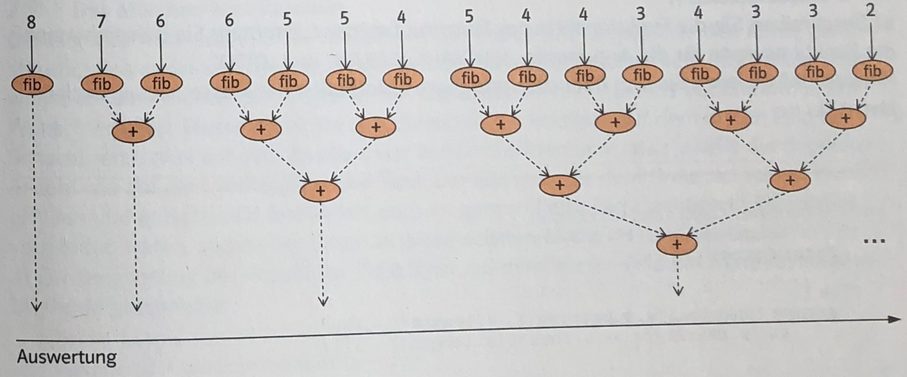
Fig. 2

Die Funktion *fib*, welche die Seitenlängen der Quadrate in den einzelnen Konstruktionsschritten berechnet, lässt sich damit rekursiv definieren:

Rekursionsvorschrift:

$$fib(n) = \begin{cases} fib(n-1) + fib(n-2), & \text{falls } n > 1 \\ 1, & \text{falls } n = 1 \\ 0, & \text{falls } n = 0 \end{cases}$$

Das dynamische Datenflussdiagramm zeigt, dass die Anzahl der Funktionsaufrufe in jedem Schritt vergrößert wird. Man spricht deshalb von einer kaskadenartigen Rekursion.



Leonardo da Pisa (1180–1241), genannt Fibonacci, war ein italienischer Mathematiker.

Rekursive Funktionen enthalten in ihrem Funktionsterm mindestens einmal den eigenen Funktionsbezeichner. Ein Funktionsaufruf endet nur dann nach einer endlichen Anzahl von Berechnungsschritten, wenn die Abbruchbedingung erfüllt ist.