Ansätze der Programmierung:
Der Collatz-Algorithmus mit Maxima

Um mit einfachen Programmierstrukturen (Schleifen und Verzweigungen) zu arbeiten, eignen sich sehr schön die Collatz-Zahlen. Den folgenden Algorithmus sollten Schüler zunächst "händisch" durchführen.

Nimm eine natürliche Zahl $n > 0$, starte erst einmal mit einer Zahl kleiner als 20.
Unterwirf deine Zahl folgendem Algorithmus (Verfahren):

- Wenn n ungerade, dann ersetze n durch $3n + 1$.
- Wenn n gerade, dann ersetze n durch $\frac{n}{2}$.

Unterwirf den neuen Wert von n demselben Verfahren.
Mache immer weiter, bis du merkst, dass nichts Neues mehr passiert.
Was stellst du fest? Nimm eine neue Zahl und prüfe, ob dasselbe Ergebnis eintritt.

Nachdem einige Beispiele mit der Hand gerechnet und verglichen wurden, wächst der Wunsch, dies zu automatisieren und zu prüfen, ob man auch für sehr große Zahlen immer auf 4-2-1-4-2-1... kommt.

Dazu muss der Algorithmus so lange durchgeführt werden, bis sich eine Eins ergibt, also so lange die Zahl größer als Eins ist: `while` (Schleife). Bei jedem Schritt muss geprüft werden, ob die Zahl gerade oder ungerade ist: `oddp(i):` Prüfung, ob i ungerade. In Abhängigkeit davon muss dann i durch 3i+1 oder i/2 ersetzt werden (Verzweigung: `if oddp(i) then i:3*i+1 else i:i/2`). Für jeden Schritt soll das Ergebnis angezeigt werden: `display(i)` innerhalb der while-Schleife.

```
(%i1)  i:1001$
       while i>1 do ( [if oddp(i)  then i:3*i+1  else i:i/2 ] , display(i) );
```

$i = 3004$
... 140 Zeilen ...
$i = 1$

(%o2) *done*

Zählen der Schritte

Nach einigen Versuchen interessiert man sich nicht mehr für die Zwischenergebnisse, sondern nur noch dafür, wie viele Schritte man benötigt, um zur Eins zu gelangen. Dazu kann man eine Variable einführen, die bei jedem Durchlaufen der Schleife um Eins hochgezählt wird (`k:k+1`). Deren Wert wird am Ende mittels `return(k)` zurückgegeben. Die im Folgenden beschriebene Funktion ***collatz(n)*** gibt also die Anzahl der Schritte des Collatz-Algorithmus bis zum Ende (Erreichen der Zahl 1) bei der Anfangszahl n aus:

```
(%i3)  collatz(n):=block( [i:n, k:0],
                  while i>1 do ([if oddp(i) then i:3*i+1 else i:i/2], k:k+1),
                  return(k) )$

(%i4)  collatz(1001);
```

(%o4) 142

Zeichnen eines Diagramms
(Anzahl der Schritte des Collatz-Algorithmus in Abhängigkeit vom Eingabewert)

Es ergibt sich ein faszinierendes Muster, wenn man die Anzahl der Schritte des Collatz-Algorithmus in Abhängigkeit vom Eingabewert ausgibt. Dazu erstellt man eine Liste aus den Wertepaaren [i,collatz(i)] für i von 1 bis zu einem Endwert (unten sind dafür 5000 angegeben) und zeichnet die entsprechenden Punkte in ein Diagramm.

```
(%i5)  load(draw)$
       draw2d(point_type=7, point_size=0.25,
              points(makelist([i,collatz(i)],i,1,5000)))$
```