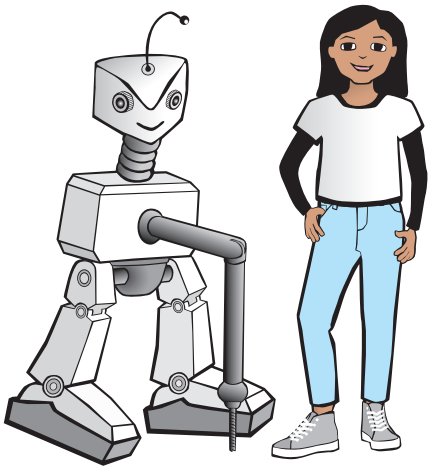
Informatik, fertig, los! (Zusatzmaterial)

Manual für Lehrpersonen

Petra Adamaszek, Bernd Gärtner, Ivana Klasovita



Eine freundliche Bitte zuerst...

Liebe Leserin, lieber Leser – Herzlich willkommen!

Wir freuen uns, Ihnen das Online-Zusatzmaterial zur Lern-Kiste "Informatik, fertig, los!" aus dem Kinderlabor vorzustellen. Dieses Dokument ist das Manual für Lehrpersonen zum Zusatzmaterial. Sie finden alle Materialien zu "Informatik, fertig, los!" auch elektronisch auf der Webseite des Kinderlabors, oder über diesen QR-Code:

Wir empfehlen Ihnen, zuerst das Manual zum Material zu lesen, das der Kiste beiligt. Das Zusatzmaterial besteht aus einem Extra-Arbeitsheft (mit zugehörigem Lösungsheft), das für stärkere SuS oder in der Begabtenförderung eingesetzt werden kann.

Wir wünschen viel Spass und Lernerfolg!

Inhaltsverzeichnis

4 Weite Wege **2**
4.1 Überblick 2
4.2 Didaktische Hinweise 3
4.3 Hinweise zu den Aufgaben 6

4 Weite Wege

4.1 Überblick

In diesem Abschnitt betrachten wir nur noch Programme mit Wiederholungsbefehlen sowie dem Befehl $\boxed{+1}$. Mit einem solchen Programm geht Robo einfach eine bestimmte Anzahl von Feldern vorwärts (macht eine bestimmte Anzahl von Schritten). Die SuS werden sehen, dass bereits mit sehr kurzen Programmen sehr viele Schritte gemacht werden können, und wie systematisch Programme gebaut werden können, mit denen Robo eine vorgegebene Anzahl von Schritten macht.

In diesem Teil geht es sehr viel um Mathematik. Relevante Lernziele betreffen Addition, Multiplikation, Division (mit Rest). Es geht aber auch um Informatik-Kompetenzen, die offiziell aber erst nach dem Orientierungspunkt im Zyklus 2 kommen:

MI 2.2: Die SuS können Programme mit Schleifen, bedingten Anweisungen und Parametern schreiben und testen.

Insbesondere das Konzept der Funktionen (oder Unterprogramme) wird hier thematisiert. Es geht darum, zu zeigen, wie mit wenigen Befehlen gestartet werden und der Befehlssatz dann beliebig erweitert werden kann, indem Programme für bestimmte Aufgaben geschrieben und dann mit einem neuen Befehl benannt werden werden. So steht zum Beispiel der Befehl $\boxed{+10}$ für "Mache 10 Schritte!" und kann als Abkürzung überall benutzt werden, wo Robo 10 Schritte machen soll. So können durch Kombinieren von solchen Abkürzungen am Ende sehr komplexe Aufgaben gelöst werden, auf die man ohne Abkürzungen nicht einfach kommen würde.

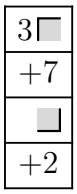
4.2 Didaktische Hinweise

Zunächst greifen Sie den vorherige Abschnitt im regulären Arbeitsheft (Wiederholungen) wieder auf.

Aufgabe 29 ist die Einstiegsaufgabe, bei der es darum geht, die Art von Programmen zu lesen, um die es in diesem Abschnitt geht, und herauszufinden, wieviele Schritte sie Robo jeweils machen lassen. Die Programme enthalten Wiederholungen, aber noch keine geschachtelten Wiederholungen.

Aufgabe 30 ist eine wichtige Aufgabe; hier geht es bereits darum, Programme zu schreiben, mit denen Robo eine gegebene (kleine) Zahl von Schritten macht. Dass die Programme möglichst kurz sein sollen (die erlaubte Anzahl von Befehlen ist vorgegeben), zwingt die SuS dazu, Wiederholungen zu benutzen. Nach dem Lösen dieser Aufgabe kennen die SuS Programme, mit denen Robo 2,3,...,10 Schritte macht. Diese werden im weiteren als Bausteine für mehr Schritte benutzt, indem sie mit neuen Abkürzungsbefehlen benannt werden.

Besprechen Sie mit den SuS Abkürzungsbefehle und Abkürzungsprogramme (Seite 3) und erarbeiten Sie mit den SuS, für wieviele Schritte das Abkürzungsprogramm

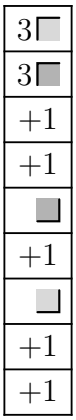


steht, nämlich 23. Wichtig ist hier, herauszuarbeiten, wo die 23 herkommt: $23 = 3 \cdot 7 + 2$. Dass also die Schrittzahl innerhalb einer Wiederholung mit der Anzahl der Wiederholungen multipliziert wird, und dass im Programm hintereinander stehende Schrittzahlen addiert werden.

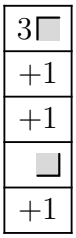
Falls die Bausteine benutzt werden, können Abkürzungsbefehle durch Überkleben der +1 mittels Post-Its erzeugt werden (zum Beispiel mit Beschriftung

+7). Erklären Sie, wie aus dem Abkürzungsprogramm ein echtes Programm (ohne Abkürzungen und Post-Its) erhalten werden kann, das für die gleiche Anzahl von Schritten steht.

Dabei entsteht, ohne dass dies gross thematisiert werden muss, eine geschachtelte Wiederholung:



Einige SuS werden dies merken und sich vielleicht wundern, was das denn zu bedeuten habe. Dann kann gesagt werden, dass der Teil



mit der inneren Wiederholung ja einfach das echte Programm für $\boxed{+7}$ ist, das hier also 7 Schritte gemacht werden. Und dass die äussere Wiederholung das dreimal wiederholt, so dass 21 Schritte gemacht werden (dann kommen noch die letzten beiden $\boxed{+1}$ dazu, so dass wir auf 23 kommen).

Schliesslich sagen Sie, dass nun auch die Abkürzung $\boxed{+23}$ benutzt werden darf (eine Abkürzung darf erst dann benutzt werden, wenn ein echtes Programm dafür gebaut wurde).

Ab Aufgabe 35 (Seite 7) geht es darum, mit existierenden Abkürzungen neue Abkürzungen zu bauen, mit denen dann wiederum neue Abkürzungen gebaut werden, und so weiter. Dieser *induktive* Aufbau ist sehr typisch für die Informatik, insbesondere fürs Programmieren: angefangen wird mit kleinen Programmen; diese kombiniert man auf einfache Weise zu grösseren, und diese dann wiederum zu noch grössseren, bis sich am Ende ein komplexes Programm ergibt, das überraschend viel kann. Dabei war jeder einzelne Schritt ganz einfach.

Aufgabe 35 zeigt, wie dieses schrittweise Vorgehen am Ende zu Programmen für viele Schritte führt. Hier ist es wichtig, das Beispiel genau durchzugehen, damit die SuS wissen, wie die Aufgabe gemeint ist, und was sie genau machen sollen.

Mathematisch passiert hier folgendes: wir bauen grosse Zahlen, indem wir mit kleinen Zahlen beginnen, diese mit Hilfe einfacher Multiplikationen und Additionen zu grösseren Zahlen kombinieren, diese wiederum zu noch grössseren, bis sich am Ende eine wirklich grosse Zahl ergibt.

In der Informatik ist dies auch als *bottom-up*-Ansatz bekannt (Erstellen einer Lösung aus existierenden Lösungen für Teilprobleme).

Der nächste Theorieteil (ab Seite 9) erfordert Kenntnis der Division mit Rest. Hier geht es darum, Programme zu bauen, mit denen Robo genau eine (grosse) vorgegeben Anzahl von Schritten machen kann. Der Ansatz besteht darin, das vorherige Vorgehen (schrittweises Bauen grössser Abkürzungen aus kleineren) umzudrehen und grössere Abkürzungen schrittweise auf kleinere zurückzuführen, bis wir bei Abkürzungen ankommen, für die wir schon echte Programme gebaut haben. Dies ist der *top-Down*-Ansatz der Informatik: Wiederholte Reduktion eines Problems auf Teilprobleme, bis Teilprobleme erreicht sind, die einfach zu lösen sind.

Erarbeiten Sie das Vorgehen mit den SuS am Beispiel der Zahl von 70 Schritten. Per Division mit Rest erhalten wir $70 = 4 \cdot 17 + 2$, also haben wir das Problem für 70 Schritte auf das Problem für 17 Schritte reduziert. Entsprechend verfahren wir dann mit der Zahl 17 ($17 = 4 \cdot 4 + 1$), also haben wir das Problem für 17 Schritte auf das Problem für 4 Schritte reduziert, wofür wir bereits eine Lösung haben. Indem wir nun genau wie im Abschnitt zuvor vorgehen, können wir dann die echten Programm für 17 und 70 Schritte bauen.

Statt jeweils mit Rest durch 4 zu teilen, könnten wir genauso auch durch 3 oder 2 teilen und die entsprechenden Wiederholungsbefehle benuzten. Hier kann thematisiert werden, warum 4 generell besser ist (die Zahlen werden schneller kleiner, und wir erreichen deshalb schneller Abkürzungen, für die wir bereits Programme kennen). Allerdings ist Division durch 4 nicht immer besser (hier kann das Beispiel $\boxed{+18}$ aus Aufgabe 33 besprochen werden).

Ziel am Ende des Zusatzmaterials ist es, dass SuS für beliebige Zahlen (im Zahlenraum bis 1000) auf systematische Weise (unter Benutzung von Division mit Rest und Abkürzungsbefehlen) Programme bauen können, mit denen Robo die entsprechende Anzahl von Schritten geht. Es geht dabei nicht darum, dass die Programme kürzestmöglich sind, obwohl einige Knobelaufgaben in diese Richtung gehen. Tatsächlich kann es interessant sein, aus der Suche nach einem möglichst kurzen Programm für eine gegebene Schrittzahl einen Wettbewerb zu machen.

Letztlich sind Robo und seine Programmierung “nur” ein Aufhänger, Mathematik zu machen. Die mathematische Aufgabe besteht hier darin, eine grosse Zahl “effizient” (mit einem kurzen Term) als Produkt und Summe von kleinen

Zahlen zu schreiben, wobei als Multiplikatoren (von links) nur 2, 3 und 4 erlaubt sind, und als Summanden nur Einsen. Natürlich kann jede Zahl n als Summe von n Einsen geschrieben werden, das ist aber nicht effizient. Mit Hilfe von Multiplikationen kann die Termlänge reduziert werden. Beispiele:

$$\begin{aligned} 16 &= 4 \cdot (4 \cdot 1), \\ 18 &= 3 \cdot (3 \cdot (1 + 1)) \\ 149 &= 4 \cdot (4 \cdot (4 \cdot (1 + 1) + 1) + 1) + 1 \end{aligned}$$

Als reine Mathematikaufgaben wären solche Aufgaben sehr künstlich, aber im Rahmen der Programmierung ergeben sie für die SuS Sinn.

4.3 Hinweise zu den Aufgaben

Aufgabe 31 übt das Lesen von Abkürzungen; in jedem Fall soll die Schrittzahl als Ergebnis von passenden Multiplikationen und Additionen hergeleitet werden.

Aufgabe 32 vertieft das und übt auch das Bauen der echten Programme hinter den Abkürzungsprogrammen; hier sollen die SuS ihre in Aufgabe 30 gebauten echten Programme für die Abkürzungen einsetzen. Sie sollen erkennen, dass die Länge der resultierenden echten Programme auch davon abhängt, mit welchen Abkürzungen wir anfangen.

In Aufgabe 33 soll nun ein möglichst kurzes echtes Programm mit einer gegebenen Schrittzahl von 18 gebaut werden. Die vorherige Aufgabe hat bereits darauf vorbereitet, nun muss aber ein Abkürzungsprogramm zum ersten Mal selbst *geschrieben* und nicht nur gelesen werden.

Aufgabe 34 geht in die gleiche Richtung, allerdings werden Abkürzungen nicht erwähnt, was die Aufgabe offener macht. Die SuS können und sollen natürlich trotzdem wie vorher mit Abkürzungen anfangen. Sie erkennen hier, dass die Abkürzung $\boxed{+3}$ auf zwei Arten mit 3 echten Befehlen realisiert werden kann.

In Aufgabe 36 sollen die echten Programme zu den Abkürzungsprogrammen in Aufgabe 35 gebaut werden. Die SuS haben bereits vorher gesehen, wie eine Abkürzung durch ein echtes Programm ersetzt werden kann, und das sollen sie hier mit den Abkürzungsprogrammen in Aufgabe 35 machen. Da die Ersetzung in mehreren Schritten stattfindet, ist das richtige Vorgehen den SuS hier vielleicht nicht klar, und es gibt tatsächlich zwei grundsätzlich verschiedene Möglichkeiten.

Die empfohlene (weil auch später benutzte) Möglichkeit besteht darin, die Teilaufgaben jeweils von *rechts* nach *links* durchzugehen, und im aktuellen Programm die Abkürzungsbefehle jeweils gemäss dem letzten Schritt zu ersetzen, bis sich am Ende ein echtes Programm ergibt (wie im Theorieteil auf Seite 10 zum Beispiel für $\boxed{+17}$ gezeigt). Das entspricht dem vorher erwähnten Top-Down-Ansatz.

Es kann aber auch von links nach rechts gearbeitet werden, so wie es dem Fluss der Aufgabe 35 enstpricht, indem im jeweils nächsten Schritt das im vorherigen Schritt erhaltene echte Programm in den Platzhalter des nächsten Programms eingesetzt wird (Achtung: in (b) muss das echte Programm in einem Schritt *zweimal* eingesetzt werden). Das ist der Bottom-Up-Ansatz.

Welchen Ansatz die SuS hier bevorzugen, spielt keine Rolle, solange das Verfahren korrekt umgesetzt wird. Später wird sich der Top-Down-Ansatz als praktischer erweisen (siehe Aufgabe 37).

In beiden Fällen handelt es sich um einen rein mechanischen Prozess, der am Ende aber zu korrekten Programmen mit mehrfach geschachtelten Wiederholungen führt, die auf direkte Weise (also ohne Abkürzungen) nur sehr schwer zu konstrieren wären. Wichtig ist hier, zu thematisieren, dass die resultierenden echten Programme für Robo gut ausführbar sind, für uns Menschen aber nicht besonders lesbar. Was aber kein Problem darstellt, denn die "lesbaren Varianten" dieser Programme sind die Abkürzungsprogramme, aus denen sie entstanden sind.

Aufgabe 37 übt die gerade besprochene Theorie ein (Erstellung eines Programms mit einer vorgegebenen Schrittzahl), und in Aufgabe 38 werden wie zuvor die echten Programme dazu gebaut. Hier ist der Top-Down-Ansatz (beginne mit dem zu lösenden Problem und reduziere es schrittweise auf einfacherere Probleme) das Verfahren der Wahl.

In Aufgabe 39 geht es explizit darum zu erkennen, dass Division durch 4 nicht immer zum kürzesten echten Programm führt. Weil 27 eine Dreierpotenz ist, ist es hier besser, jeweils (ohne Rest) durch 3 zu teilen, weil dann keine Extra-Befehle ausserhalb von Wiederholungen benötigt werden.

Aufgabe 40 können Sie zu einem Wettbewerb machen, indem Sie eine Zahl vorgeben, die (möglichst kurz) "gebaut" werden soll.