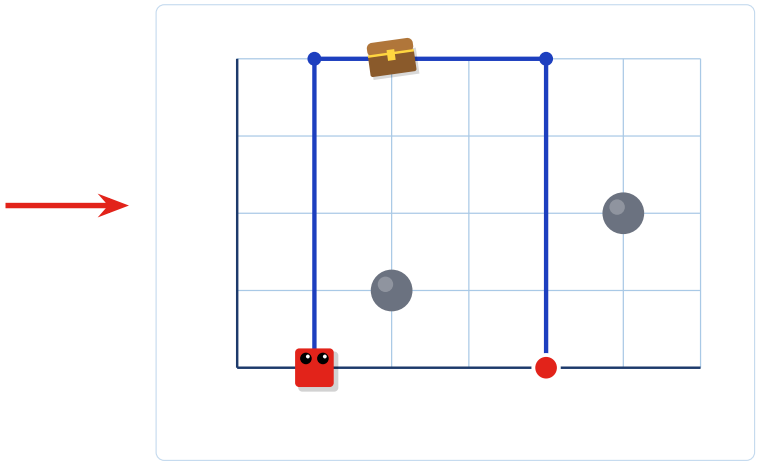
Programmierhandbuch für Aleph.

Algebra **e**ntdecken, **p**rogrammieren und **h**erleiten

```
a <- 4
vorwärts(a)
rechts()
b <- 3
vorwärts(b)
rechts()
vorwärts(a)
```



Willkommen in der Welt der Programme! Mit der Programmiersprache Aleph kannst du einen Meeresroboter programmieren, der nach Schätzen sucht. Wir können auf zwei verschiedene Weisen Aleph programmieren: zum einen auf Papier, mit echten Variablenschachteln und einem Modellroboter aus Holz, zum anderen mit dem Programm Aleph am Computer.

In diesem Handbuch gibt es viele Aufgabenblätter. Du brauchst die Aufgabenblätter nicht nacheinander durchzuarbeiten, aber sie sind so gemacht, dass sie am Anfang leichter und nach hinten hin schwerer werden. Such dir die Aufgaben aus, die du machen möchtest.

Inhalt

1 Programmiersprache 2

2 Bevor es losgeht 3

3 Einführung 5
- 3.1 Erste Aufgaben mit dem vorwärts-Befehl . . . 5
- 3.2 Puzzle zum vorwärts-Befehl . . . 15
- 3.3 Die „für“-Schleife und der „geheZumFeld“-Befehl . . . 22
- 3.4 Puzzle zum geheZumFeld-Befehl und zur für-Schleife . . . 26

4 Forschung 32
- 4.1 Verborgene Schätze . . . 37
- 4.2 Mathematisches . . . 38

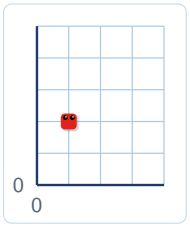
5 Über Aleph 41

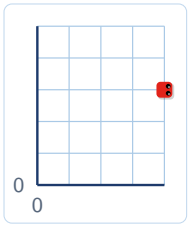
© 2007–2026 Annika Wille

1 Programmiersprache

Der Roboter

Der Meeresroboter fährt oder schwimmt über dem Meeresboden. Auf dem Meeresboden denken wir uns ein Gitter. Dabei kann der Roboter immer auf einem Schnittpunkt stehen. Beispiele:





Der Roboter kann in vier verschiedene Richtungen gucken:



oben rechts unten links

Überblick über die Befehle

`n <- 2`	Variablenzuweisung. „n“ ist der Name für eine Variablenschachtel (zum Beispiel eine Streichholzschachtel), „2“ oder eine andere Zahl steht für die Anzahl der Hölzer, die in der Schachtel sind.
`vorwärts(n)`	Der Roboter bewegt sich so viele Schritte vorwärts, wie Hölzer in der Schachtel sind.
`links()`	Der Roboter dreht sich um 90° nach links.
`rechts()`	Der Roboter dreht sich um 90° nach rechts.
`umdrehen()`	Der Roboter dreht sich um 180°.
`geheZumFeld(x,y)`	siehe Kapitel 3
`gleite(x,y)`	siehe Kapitel 4
`für-Schleife`	siehe Kapitel 3
`# Text`	Ein Kommentar: Alles hinter # gehört nicht zum Programm.

Zu den ersten beiden Befehlen

Bei der Variablenzuweisung muss man nicht zwingend eine Variablenschachtel „n“ nennen. Auch die Anzahl der Hölzer kann frei gewählt werden. Mögliche andere Beispiele sind:

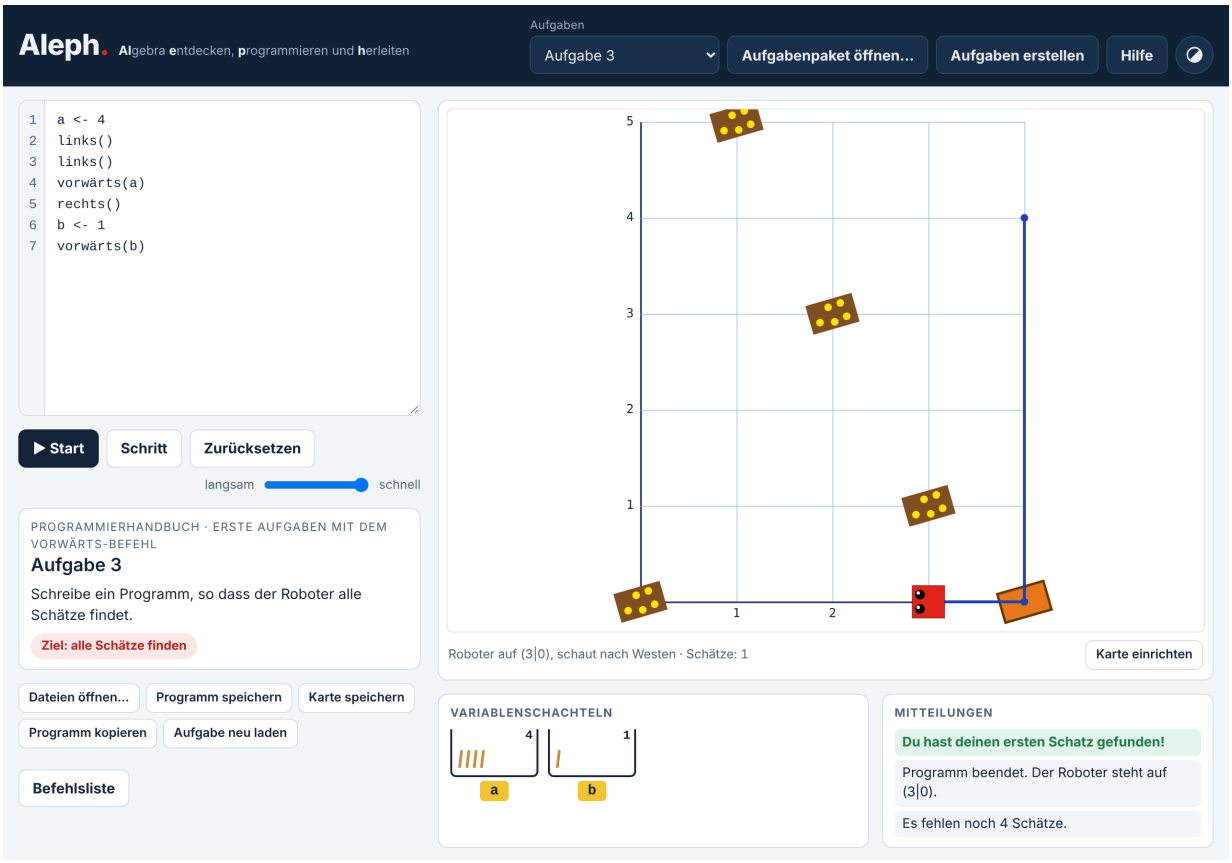
```
a <- 4                    oder:    x <- 12
vorwärts(a)                        vorwärts(x)
```

Achtung: In den Klammern von `vorwärts` muss immer eine Variablenschachtel stehen. `vorwärts(3)` geht nicht.

2 Bevor es losgeht

Das Programm Aleph

Das Programm Aleph läuft im Browser. Du musst nichts installieren. Deine Lehrerin oder dein Lehrer sagt dir, wo du es findest, zum Beispiel auf einer Internetseite oder als Datei `index.html`.



- **Links oben** schreibst du dein Programm.
- Mit **Start** läuft dein Programm. Mit **Schritt** läuft es Befehl für Befehl. **Zurücksetzen** stellt den Roboter wieder an den Anfang.
- **Rechts** siehst du die Karte mit dem Roboter. Darunter stehen die **Variablenschachteln** mit ihren Hölzern und die **Mitteilungen**.
- Unter dem Programm steht die **Aufgabenstellung**. Die Aufgaben aus diesem Handbuch findest du oben unter **Aufgaben**.
- Die **Befehlsliste** zeigt alle Befehle. Ein Klick auf einen Befehl schreibt ihn in dein Programm.

© 2007–2026 Annika Wille

Die erste Aufgabe

Schreibe ein Programm, so dass der Roboter drei Mal mit n vorwärts geht, sich zweimal dreht und dreimal mit k vorwärts geht. Finde verschiedene Zahlen für n und k, so dass der Roboter wieder zu seinem Ausgangspunkt zurückkommt.

Mit der Programmkontrolle links kannst du dein Programm starten.

Variation: Schreibe andere Programme, die im Unterricht vorkamen und an die du dich erinnerst.

3 Einführung

Manche von den Aufgaben kann man nur ohne den Computer lösen, manche nur mit dem Computer und manche auf beide Weisen.

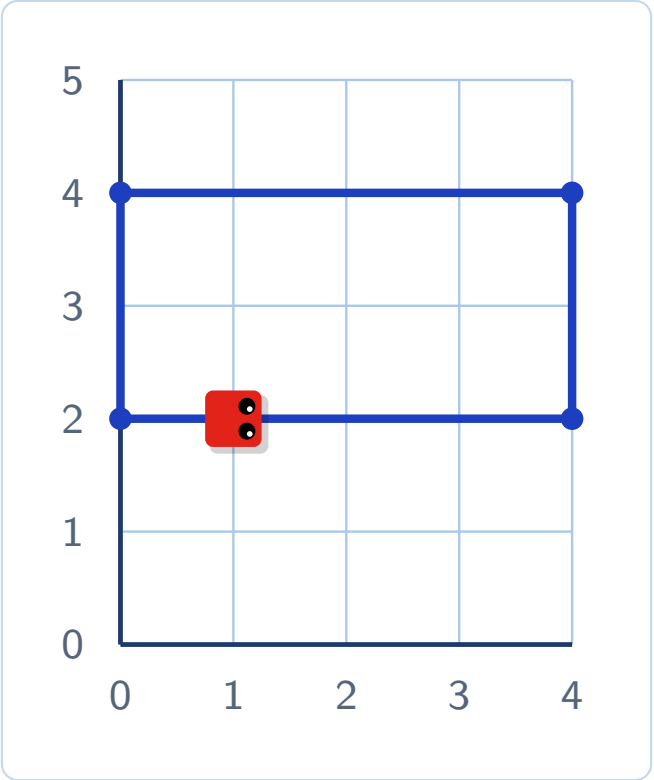
Die zur Aufgabe gehörige Karte und die Roboterposition lädt das Programm, wenn du oben unter **Aufgaben** die Aufgabe auswählst. Bei Aufgabe 1 wählst du also „Aufgabe 1“.

3.1 Erste Aufgaben mit dem vorwärts-Befehl

Bei den nächsten Aufgaben geht es immer um die Variablenzuweisung und den vorwärts-Befehl.

Aufgabe 1

Schreibe ein Programm, so dass der Roboter den folgenden Weg abfährt:

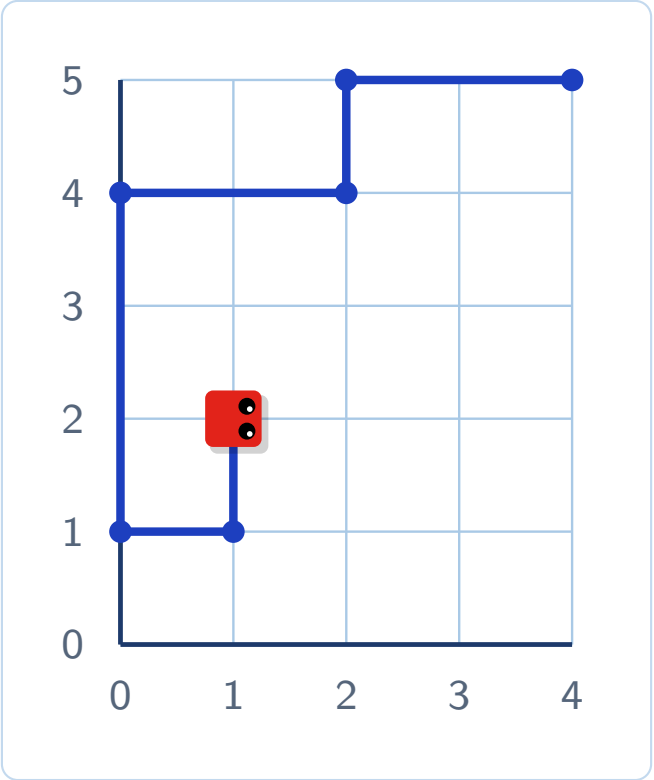


Programm:

Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 1 **Geschafft!** ☐

Aufgabe 2

Schreibe ein Programm, so dass der Roboter den folgenden Weg abfährt:



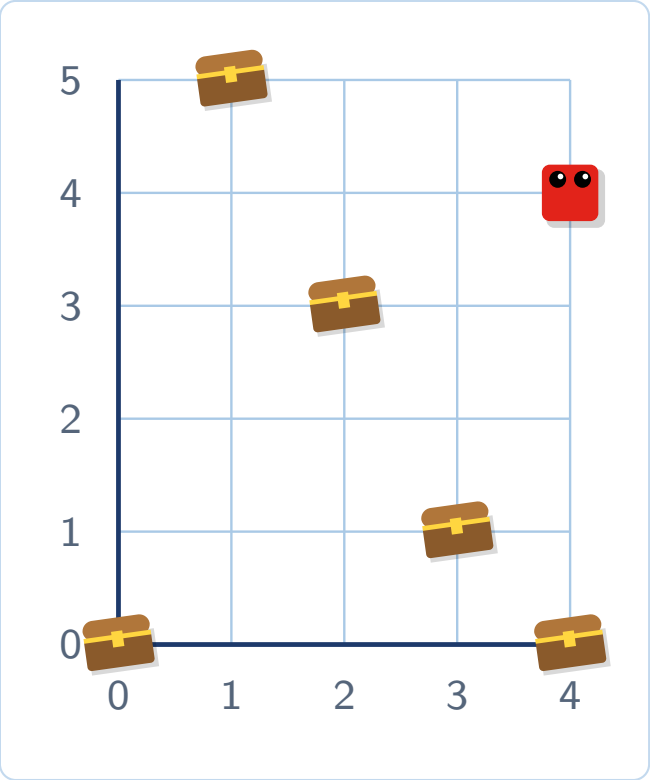
Programm:

Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 2 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 3

Schreibe ein Programm, so dass der Roboter alle **Schätze** findet.

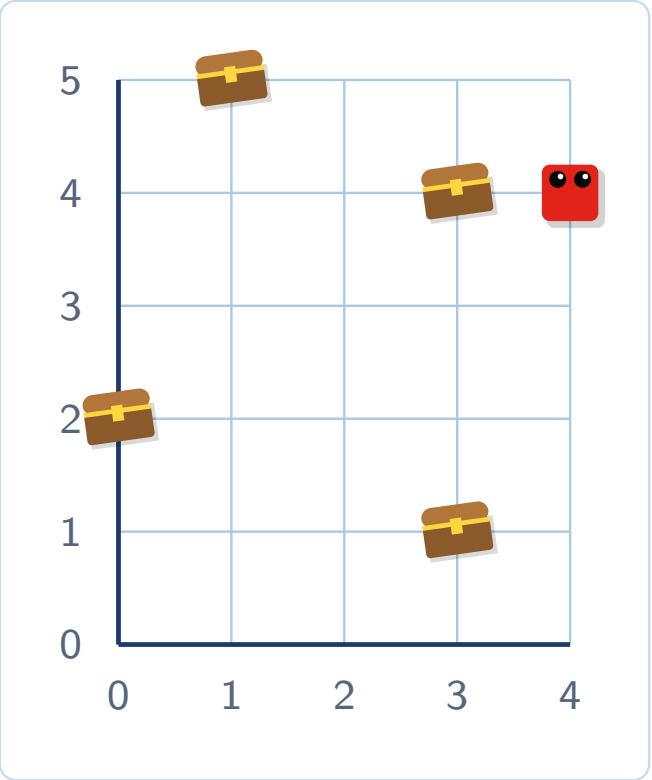


Programm:

Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 3 **Geschafft!** ☐

Aufgabe 4

Schreibe ein Programm, so dass der Roboter alle **Schätze** findet.



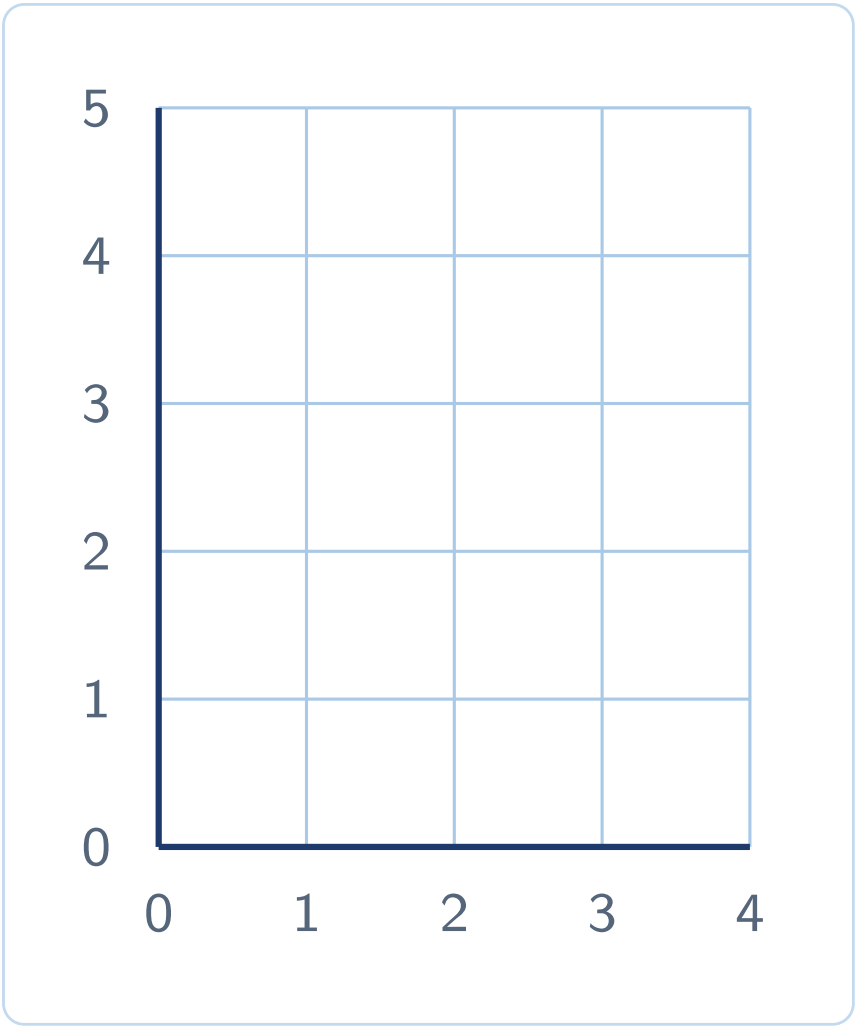
Programm:

Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 4 **Geschafft!** ☐

Aufgabe 5

Ein Mitglied deiner Gruppe zeichnet nun **Schätze** und einen Roboter in die Karte ein. Das andere oder die anderen programmieren den Roboter. Danach tauscht ihr.

Am Computer: Mit **Karte einrichten** unter der Karte setzt ihr den Roboter und die **Schätze**.

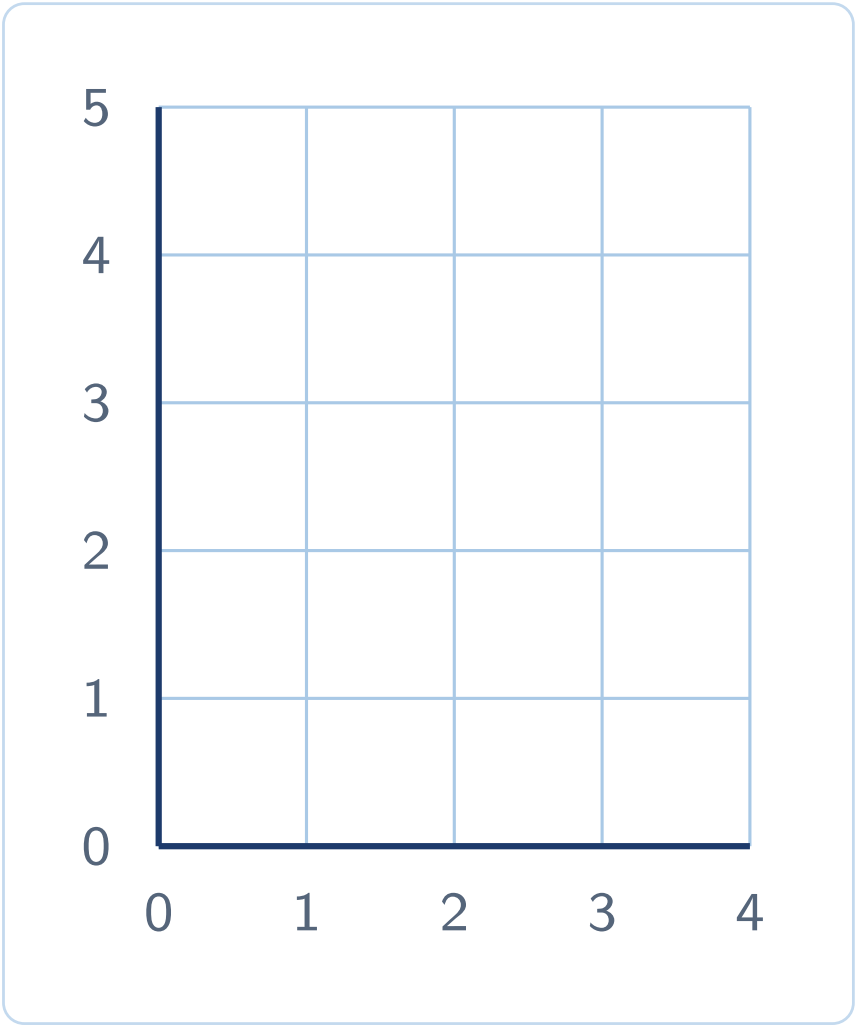


Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 5 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 6

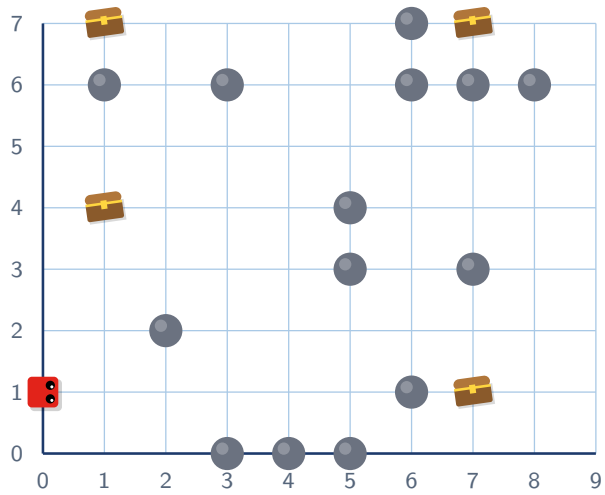
Ein Mitglied deiner Gruppe zeichnet einen genauen Weg des Roboters in die Karte ein. Dicke Punkte bedeuten, dass der Roboter dort mit einem Befehl ankommt. Das andere oder die anderen programmieren den Roboter, so dass er den Weg abfährt. Danach tauscht ihr.



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 6 **Geschafft!** ☐

Aufgabe 7

Programmiere ein möglichst kurzes Programm, so dass der Roboter alle **Schätze** findet. Benutze nur die **Variablenzuweisung** und den `vorwärts`-Befehl. Die dicken grauen Kreise sind **Hindernisse**, auf die der Roboter nicht ziehen kann. Mit dem `vorwärts`-Befehl kann man die **Hindernisse** auch nicht überfahren.



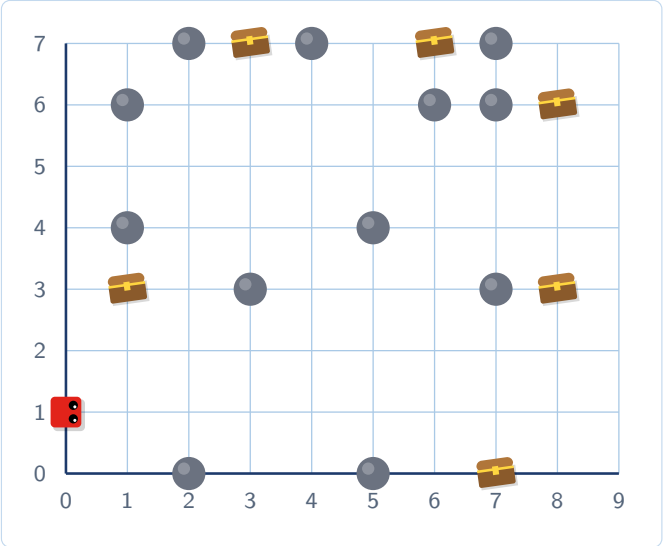
Programm:

Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 7 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 8

Programmiere ein möglichst kurzes Programm (wie in Aufgabe 7), so dass der Roboter alle **Schätze** findet. Benutze nur die **Variablenzuweisung** und den `vorwärts`-Befehl.



Programm:

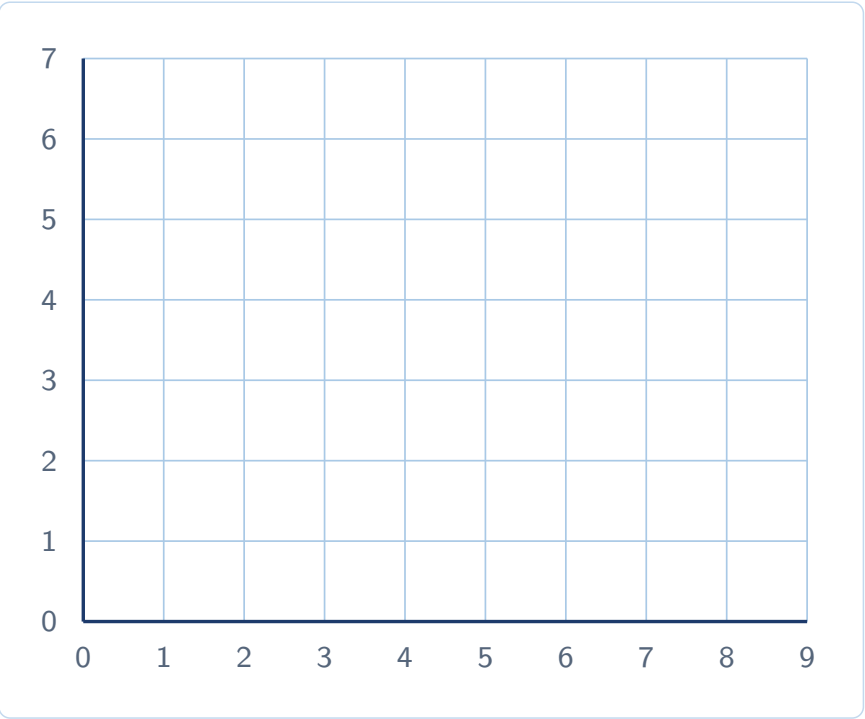
Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 8 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 9

Ein Mitglied deiner Gruppe zeichnet **Schätze** und **Hindernisse** in die Karte ein. Das andere oder die anderen programmieren den Roboter, so dass er die **Schätze** findet. Danach tauscht ihr.

Am Computer: Mit **Karte einrichten** unter der Karte setzt ihr den Roboter, die **Schätze** und die **Hindernisse**.



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 9 **Geschafft!** ☐

© 2007–2026 Annika Wille

3.2 Puzzle zum vorwärts-Befehl

Bei jedem Programm-Puzzle fehlen die Werte der Variablen. Wir kennen sonst den ganzen Programmtext und wissen, wo der Roboter am Anfang war und wo er am Ende steht. Finde richtige Werte für die Variablenschachteln heraus. (Achtung: Manchmal gibt es mehrere mögliche Lösungen.)

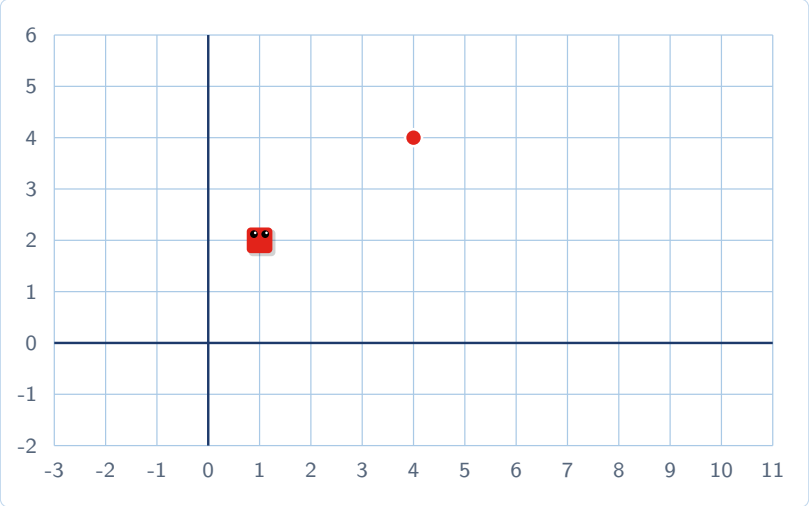
In den Bildern steht der Roboter immer auf der Startposition. Der rote runde Punkt bezeichnet den Endpunkt, also den Punkt, auf dem der Roboter nach Ablauf des Programmes steht.

Aufgabe 10

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
n <- _____
x <- _____
vorwärts(n)
rechts()
vorwärts(x)
links()
vorwärts(n)
```



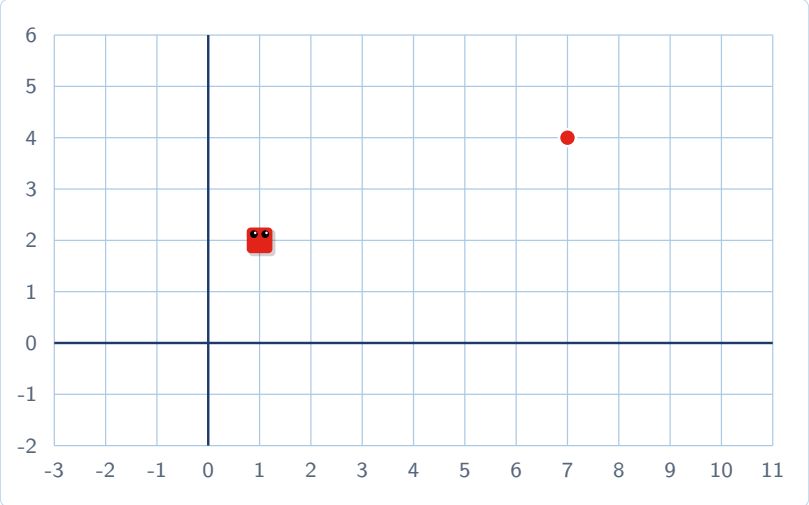
Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 10 **Geschafft!** ☐

Aufgabe 11

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
n <- _____
x <- _____
rechts()
vorwärts(n)
links()
vorwärts(x)
rechts()
vorwärts(n)
```



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 11 **Geschafft!** ☐

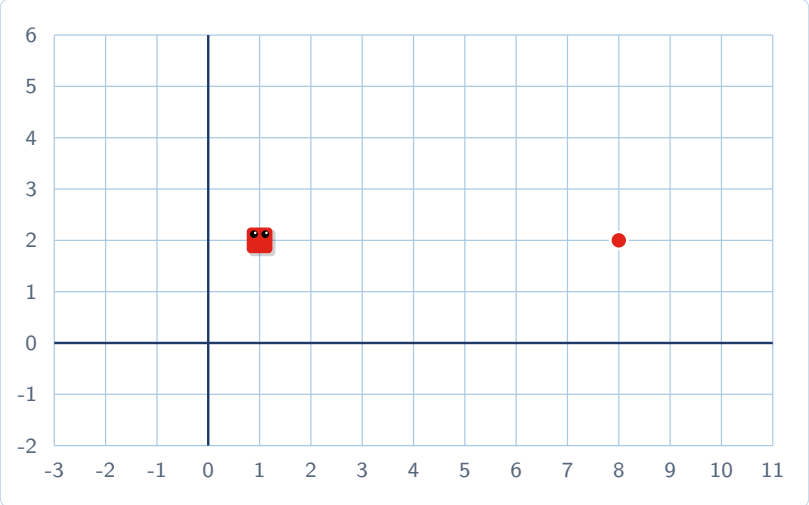
© 2007–2026 Annika Wille

Aufgabe 12

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
n <- _____
k <- _____
x <- _____
vorwärts(n)
rechts()
vorwärts(k)
rechts()
vorwärts(x)
links()
vorwärts(x)
links()
vorwärts(k)
```



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 12 **Geschafft!** ☐

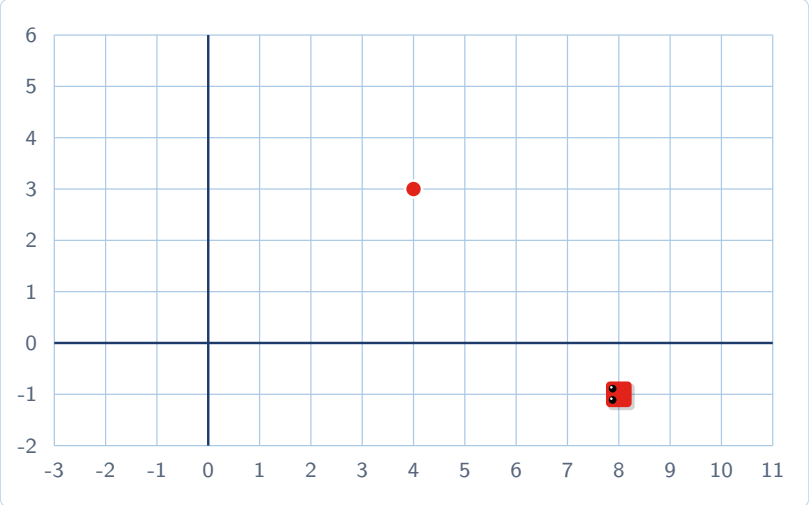
© 2007–2026 Annika Wille

Aufgabe 13

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
n <- _____
x <- _____
a <- _____
vorwärts(n)
rechts()
vorwärts(x)
rechts()
vorwärts(a)
links()
vorwärts(a)
```



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 13 **Geschafft!** ☐

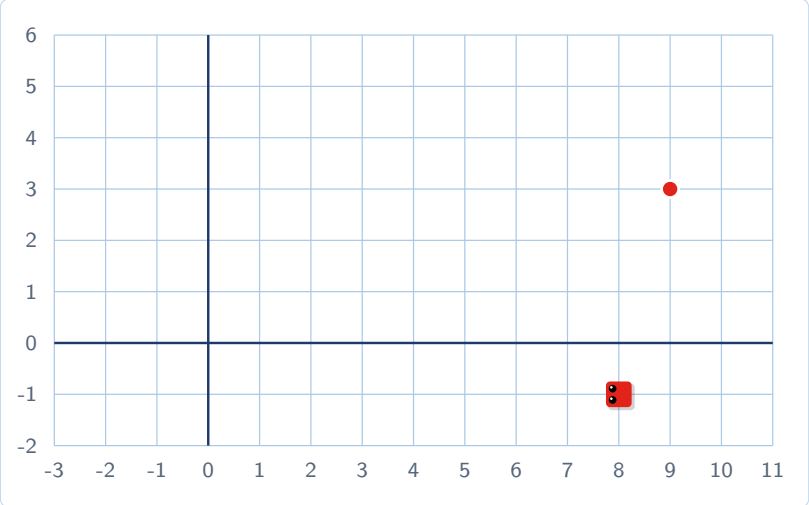
© 2007–2026 Annika Wille

Aufgabe 14

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
n <- _____
x <- _____
a <- _____
vorwärts(n)
rechts()
vorwärts(x)
rechts()
vorwärts(a)
links()
vorwärts(a)
```



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 14 **Geschafft!** ☐

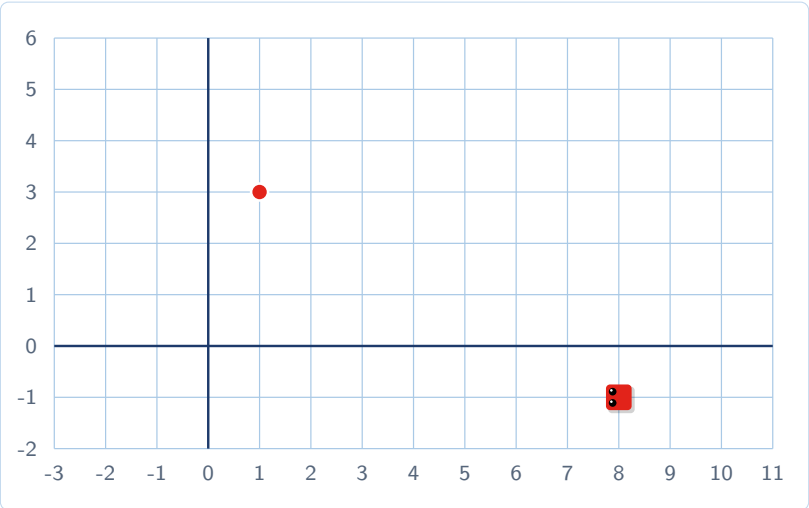
© 2007–2026 Annika Wille

Aufgabe 15

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
b <- _____
a <- _____
vorwärts(a)
vorwärts(b)
rechts()
vorwärts(a)
rechts()
vorwärts(a)
```

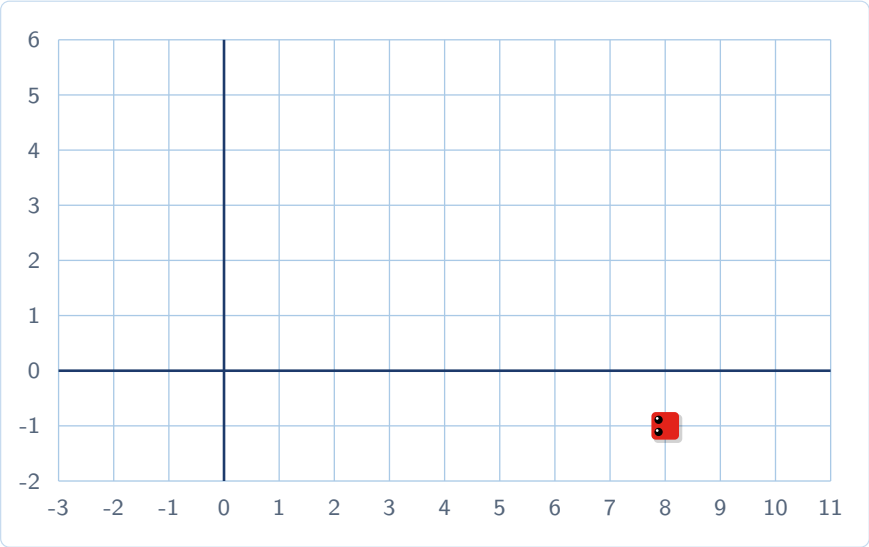


Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 15 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 16

Denkt euch selbst Programm-Puzzle aus. Ein Mitglied eurer Gruppe schreibt ein Programm und lässt dabei die **Variablenzuweisung** weg. Die anderen versuchen, mögliche Lösungen zu finden.



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 16 **Geschafft!** ☐

© 2007–2026 Annika Wille

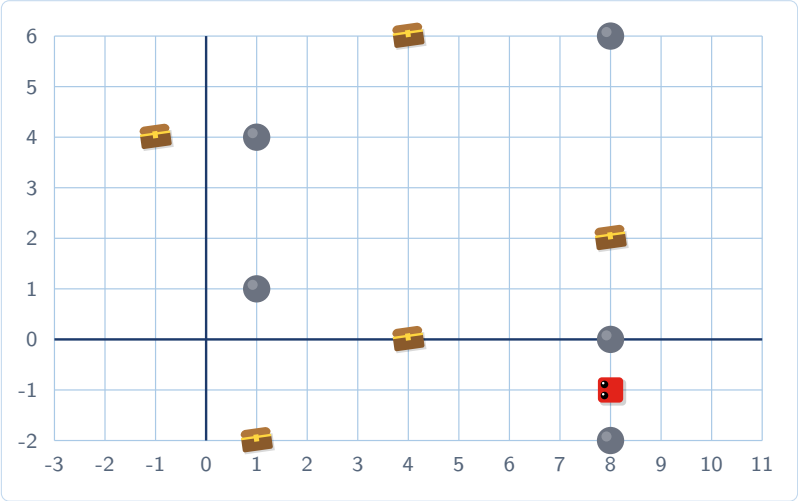
3.3 Die „für“-Schleife und der „geheZumFeld“-Befehl

Aufgabe 17

Wähle im Programm Aufgabe 17. Dort stehen schon die Karte, der Roboter und dieses Programm. Starte es und finde heraus, was die Befehle in dem Programm bedeuten.

Programm:

```
x <- 4
y <- 0
geheZumFeld(x,y)
für jedes x von 1 bis 3 mache {
  y <- 2*x
  geheZumFeld(x,y)
}
```

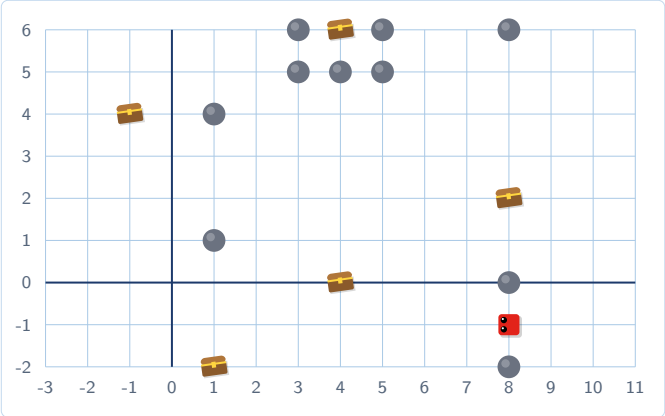


Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 17 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 18

Was ist der Unterschied zwischen dem `vorwärts`-Befehl und dem `geheZumFeld`-Befehl, wenn es darum geht, **Schätze** zu finden und über **Hindernisse** zu kommen? Wähle im Programm Aufgabe 18 und finde es heraus.



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 18 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 19

Überlege dir erst im Kopf, was die folgenden Programme machen, und probiere es dann am Computer aus.

Programm 1:

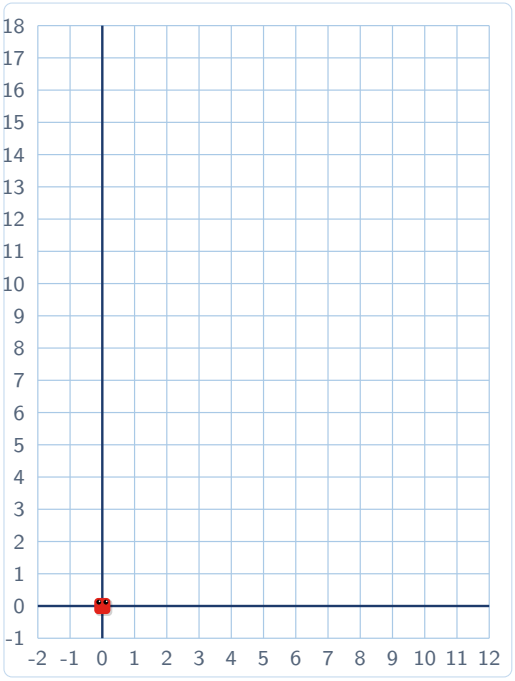
```
für jedes x von 0 bis 4 mache {
  y <- 2*x+1
  geheZumFeld(x,y)
}
```

Programm 2:

```
für jedes x von 0 bis 4 mache {
  y <- 3*x-2
  geheZumFeld(x,y)
}
```

Programm 3:

```
für jedes x von 0 bis 4 mache {
  y <- x^2
  geheZumFeld(x,y)
}
```



Die **für**-Schleifen laufen jeweils nur bis 4. Bis zu welchem Wert können sie in den Programmen höchstens laufen, so dass der Roboter nicht von der Karte fällt? Kannst du das ohne Ausprobieren herausbekommen?

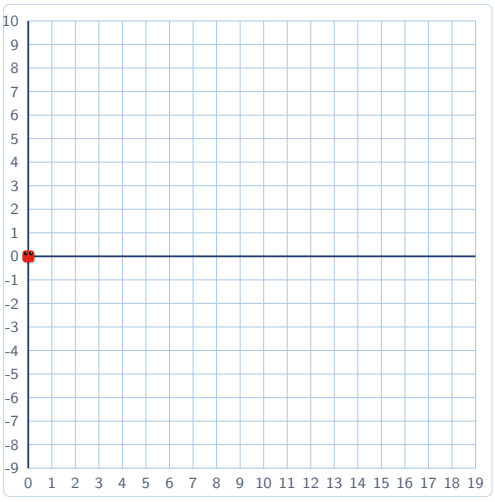
Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 19 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 20

Ein Programm lautet:

```
a <- _____
b <- _____
für jedes x von 0 bis 4 mache {
  y <- a*x+b
  geheZumFeld(x,y)
}
```



Es wurden zwei Stellen im Programmtext freigelassen. Probiere verschiedene Werte für a und b aus.

1. Was bewirken sie?
2. Beschreibe den Unterschied zwischen den **Variablenschachteln** a und b und den **Variablenschachteln** x und y.

Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 20 **Geschafft!** ☐

© 2007–2026 Annika Wille

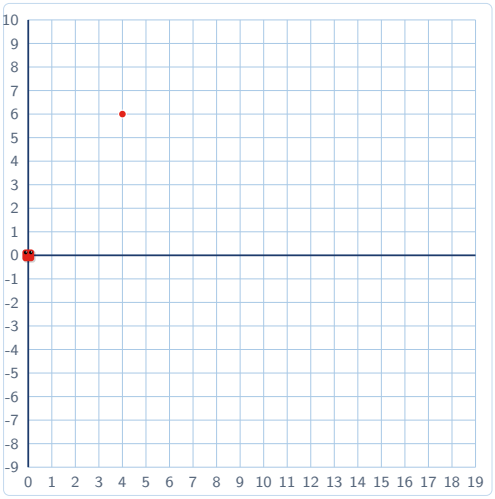
3.4 Puzzle zum geheZumFeld-Befehl und zur für-Schleife

Aufgabe 21

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
a <- _____
b <- _____
für jedes x von 0 bis 4 mache {
  y <- a*x+b
  geheZumFeld(x,y)
}
```



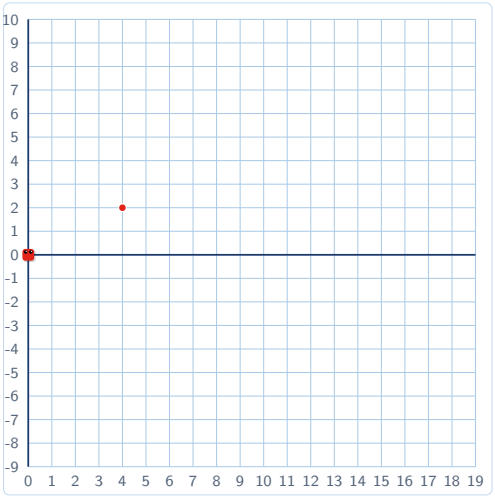
Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 21 **Geschafft!** ☐

Aufgabe 22

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
a <- _____
b <- _____
für jedes x von 0 bis 4 mache {
  y <- a*x+b
  geheZumFeld(x,y)
}
```



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 22 **Geschafft!** ☐

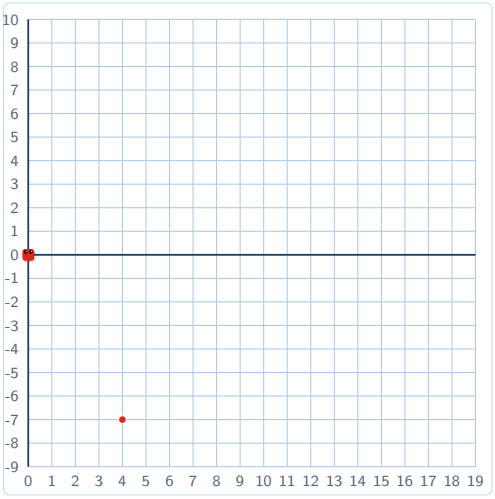
© 2007–2026 Annika Wille

Aufgabe 23

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
a <- _____
b <- _____
für jedes x von 0 bis 4 mache {
  y <- a*x+b
  geheZumFeld(x,y)
}
```



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 23 **Geschafft!** ☐

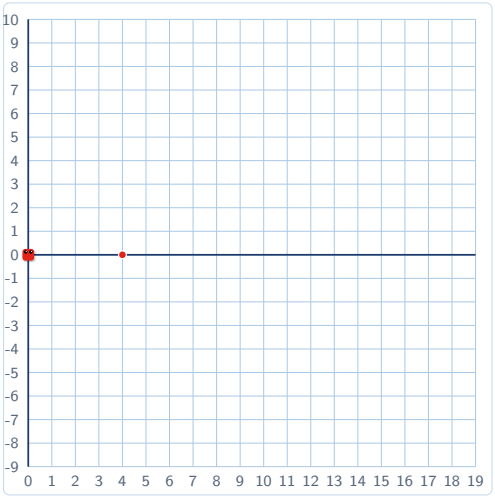
© 2007–2026 Annika Wille

Aufgabe 24

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
a <- _____
b <- _____
für jedes x von 0 bis 4 mache {
  y <- a*x+b
  geheZumFeld(x,y)
}
```



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 24 **Geschafft!** ☐

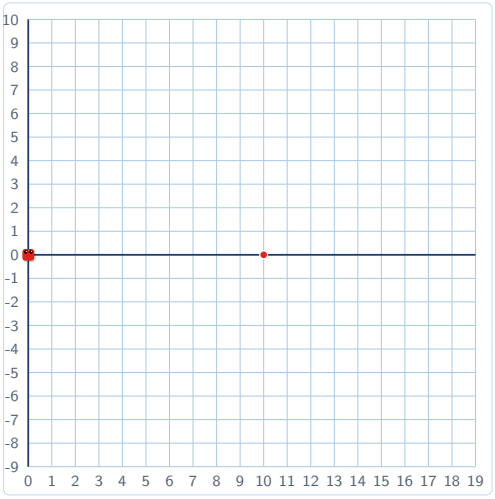
© 2007–2026 Annika Wille

Aufgabe 25

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
a <- 0
b <- _____
für jedes x von 0 bis _____ mache {
  y <- a*x+b
  geheZumFeld(x,y)
}
```



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 25 **Geschafft!** ☐

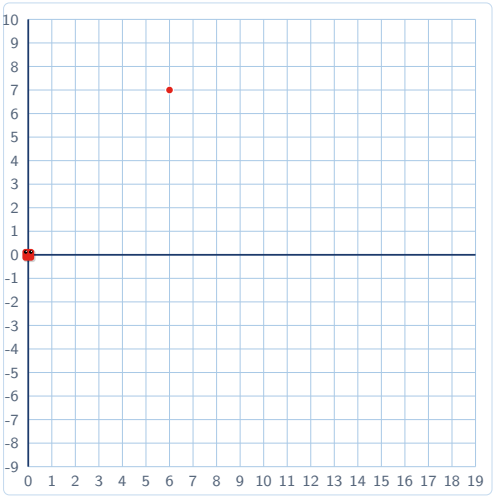
© 2007–2026 Annika Wille

Aufgabe 26

Fülle die **Lücken** im Programm, so dass der Start und **Endpunkt (der rote Punkt)** des Roboters passen. Gibt es mehrere Lösungen?

Programm:

```
a <- _____
b <- _____
für jedes x von 0 bis _____ mache {
  y <- a*x+b
  geheZumFeld(x,y)
}
vorwärts(a)
```



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 26 **Geschafft!** ☐

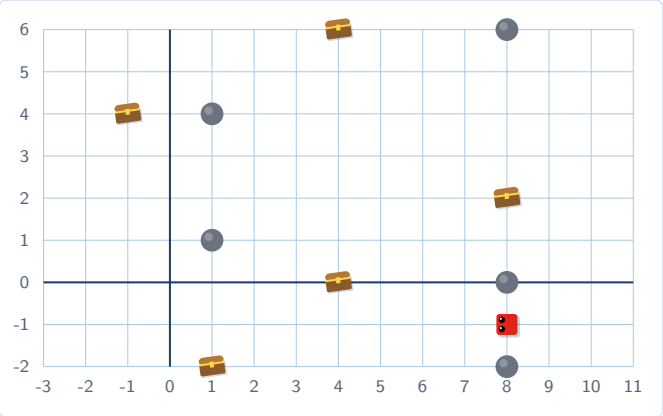
© 2007–2026 Annika Wille

4 Forschung

In diesem Kapitel sind kniffligere Aufgaben als im vorangehenden Kapitel. Es gibt immer etwas, das du herausfinden kannst. Manchmal sind die Aufgaben schwer. Lass dich also nicht entmutigen.

Aufgabe 27

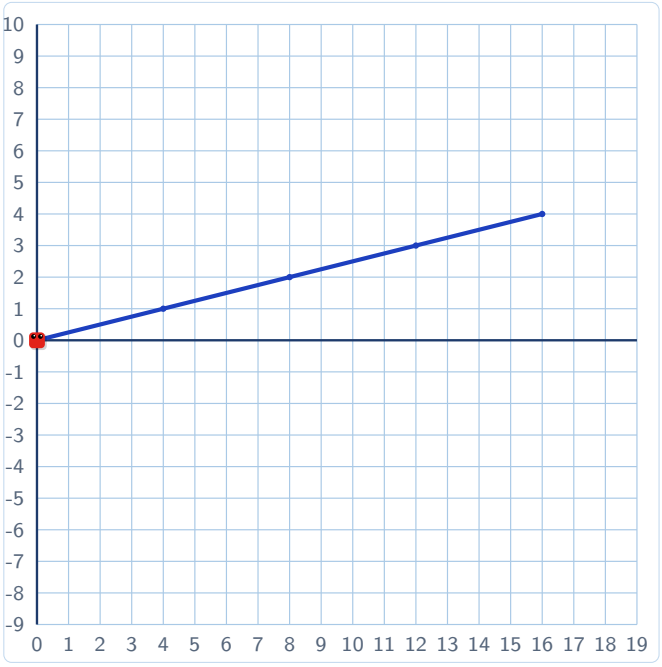
Probiere aus, was der **`gleite`**-Befehl tut. Was ist der Unterschied zwischen dem **`gleite`**- und dem **`geheZumFeld`**-Befehl?
Wähle im Programm Aufgabe 27. Dort stehen schon die Karte, der Roboter und ein Beispielprogramm. Schreibe auch eigene Programme.



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 27 **Geschafft!** ☐

Aufgabe 28

Schreibe zwei möglichst kurze Programme, so dass der Roboter genau den Weg auf dem Bild abfährt.
Bei einem Programm benutze den **`geheZumFeld`**-Befehl, beim anderen Programm den **`gleite`**-Befehl.



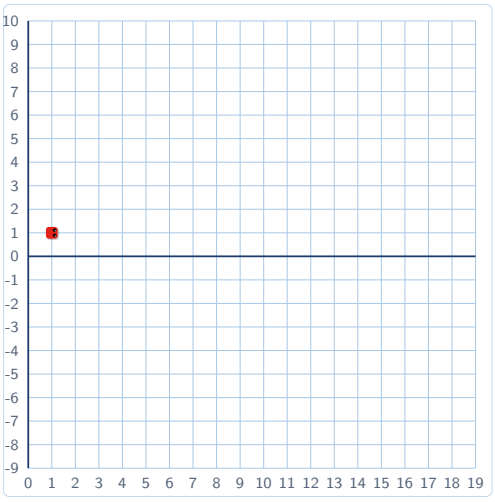
Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 28 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 29

Ein Programm sieht so aus:

```
für jedes x von 1 bis _____ mache {
  für jedes y von 1 bis x mache {
    geheZumFeld(x,y)
  }
}
```



1. Überlege dir im Kopf, was der Roboter bei verschiedenen Werten tut.
2. Probiere das Programm für verschiedene Werte aus.
3. Auf wie viele Felder kommt der Roboter, wenn man eine 1 einsetzt, oder eine 2, oder eine 3 usw.? Fällt dir etwas auf?
4. Wenn dir etwas aufgefallen ist, begründe deine Vermutung.

Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 29 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 30

Ein Roboter fährt mit dem Programm

```
für jedes x von 1 bis 5 mache {
  y <- 2*x+1
  geheZumFeld(x,y)
}
```

über eine Karte und ein anderer Roboter mit dem Programm

```
für jedes x von 1 bis 5 mache {
  y <- 3*x-2
  geheZumFeld(x,y)
}
```

1. Gibt es ein gemeinsames Feld, auf das beide Roboter kommen?
2. Wie kann man so ein Feld ohne Ausprobieren herausfinden?

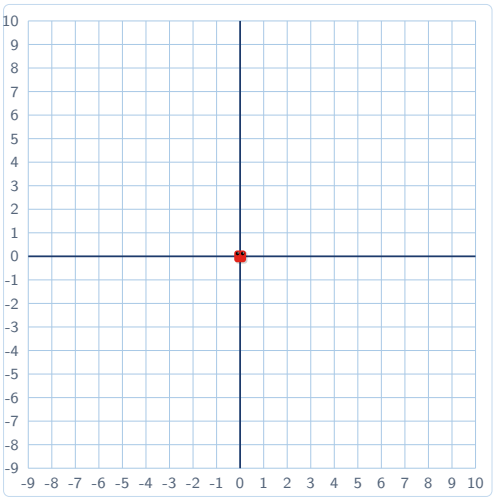
Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 30 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 31

Sieh dir dieses Programm an:

```
a <- 3
für jedes i von 1 bis 4 mache {
  vorwärts(a)
  rechts()
}
```



Mit diesem Programm fährt der Roboter ein Quadrat ab. Wenn der Roboter ein größeres Quadrat abfahren soll, muss man nur die Variablenzuweisung ändern: In der **Variablenschachtel** a soll dann eine andere Anzahl von Hölzern sein.

1. Schreibe so ein Programm für Dreiecke. Wieder soll man nur an einer **Variablenzuweisung** etwas ändern müssen, damit das Dreieck größer wird.
2. Schreibe so ein Programm für andere Vielecke (Fünfeck, Sechseck zum Beispiel).

Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 31 **Geschafft!** ☐

© 2007–2026 Annika Wille

4.1 Verborgene Schätze

In den **Aufgaben 32 bis 40** sind jeweils vier Schätze auf den Karten verborgen. Versuche, sie zu finden. Wähle dazu im Programm die Aufgabe aus.

In den **Aufgaben 41 und 42** sind Karten, die du selbst mit Schätzen versehen kannst. Wähle unter der Karte **Karte einrichten** und dann **verborgener Schatz**. Verstecke die Schätze und lass jemand anderen sie suchen.

Aufgabe	32	33	34	35	36	37	38	39	40	41/42
Erfüllt	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

4.2 Mathematisches

Aufgabe 43

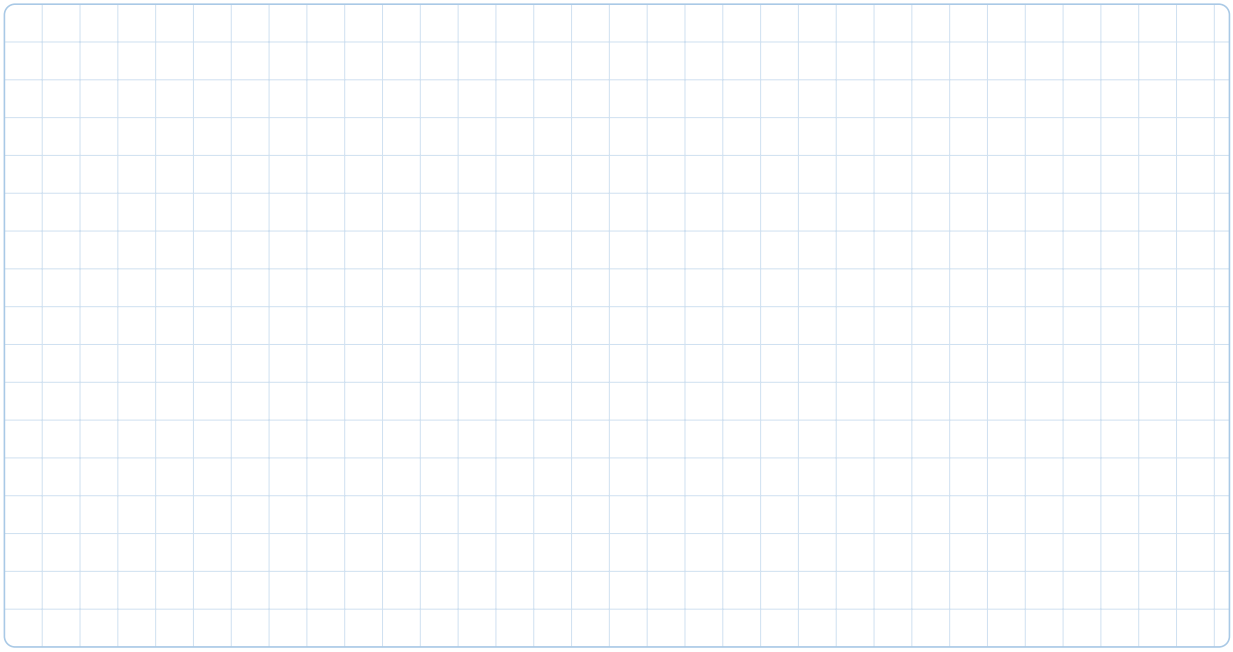
Der Roboter soll von Feld zu Feld springen. Er steht am Anfang auf der Koordinate (1,1).

Dies sind seine ersten x- und y-Werte:

x	y
1	1
4	3
7	9
10	27
13	81

1. Erkennst du Regelmäßigkeiten? Was kommt als nächste Koordinate nach (13,81)?
2. Schreibe ein Programm, so dass der Roboter diese Punkte abläuft. Es soll so sein, dass man am Anfang des Programmes angeben kann, wie weit der Roboter läuft.

Im Programm gibt es zu dieser Aufgabe zwei Roboter mit verschiedenen Startpunkten. Du wählst sie oben neben den Aufgaben aus.



Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 43 **Geschafft!** ☐

Aufgabe 44

Die y-Koordinate des Roboters bleibt immer auf 1. Die erste x-Koordinate ist 1, die zweite 1+3, die dritte 1+3+5, die vierte 1+3+5+7, usw.

1. Schreibe ein Programm, so dass der Roboter genau so läuft.
2. Fällt dir etwas bei der x-Koordinate auf?
3. Du möchtest jemand anderem erklären, was dir eben aufgefallen ist. Lass nun den Roboter so laufen, dass du es gut erklären kannst.

Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 44 **Geschafft!** ☐

© 2007–2026 Annika Wille

Aufgabe 45

In Aufgabe 29 hast du dir überlegt, auf wie viele Felder der Roboter jeweils fährt.

1. Kannst du ausrechnen, auf wie viele Felder er fährt, wenn man die 100 einträgt? Kannst du das anders ausrechnen, als nur die Zahlen nacheinander zusammenzurechnen?

Im Programm: Aufgaben → Programmierhandbuch → Aufgabe 45 **Geschafft!** ☐

© 2007–2026 Annika Wille

5 Über Aleph

Aleph steht für „**Al**gebra **e**ntdecken, **p**rogrammieren und **h**erleiten“. Die Programmiersprache und die dazu gehörende reale Lernumgebung mit Holzroboter und Streichholzschachteln wurde 2007 von Annika Wille an der Universität Bremen entwickelt. Die erste Version des Programms, Aleph 0, entstand 2007/08 in Java, bei der Thomas Ries den Parser programmierte.

Diese Browser-Fassung (2026) ist eine Neuprogrammierung von Aleph 0.

Literaturauswahl

- Wille, A. M. (2008). Aspects of the concept of a variable in imaginary dialogues written by students. *Proceedings of PME 32*, Vol. 4, 417–424.
- Wille, A. M. (2010). Steps towards a structural conception of the notion of variable. *Proceedings of CERME 6*, 659–668.
- Wille, A. M. (2011). Elementaralgebraischen Vorstellungen auf der Spur – mit selbst erdachten Dialogen. *Praxis der Mathematik in der Schule*, 53(40), 20–24.

Zu Aleph verwandte Programmiersprachen sind zum Beispiel Logo, Niki und RoboMind.

Zur Autorin

Prof. Dr. Annika Wille ist Professorin für Didaktik der Mathematik an der Leibniz Universität Hannover.

Link zur Uni-Homepage