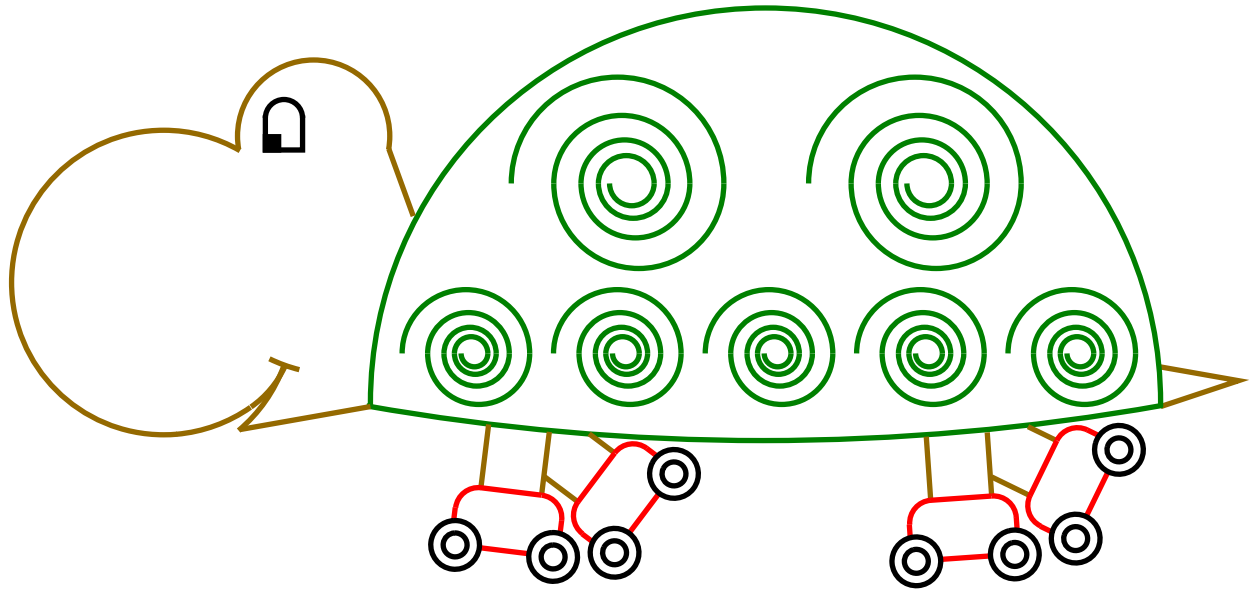
ABZ AUSBILDUNGS- UND BERATUNGSZENTRUM
FÜR INFORMATIKUNTERRICHT

Michael Barot Angélica Herrera Loyo Juraj Hromkovič

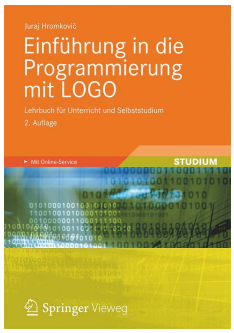
Programmieren mit LOGO



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Programmieren mit LOGO

Dieses Heft folgt im Aufbau den ersten 7 Kapiteln des Lehrbuchs *Einführung in die Programmierung mit LOGO*. Die Erklärungen wurden hier stark gekürzt oder zum Teil ganz weggelassen. Ausserdem sind einige weitere Aufgaben und Problemstellungen aufgenommen worden. Das Lehrbuch enthält viele weitere Aufgaben und Erklärungen. Ausserdem ist es mit Hinweisen für die Lehrperson versehen. Das Lehrbuch umfasst insgesamt 15 Lektionen.



Juraj Hromkovič. *Einführung in die Programmierung mit LOGO: Lehrbuch für Unterricht und Selbststudium*. 3. Aufl., Springer Vieweg 2014. ISBN: 978-3-658-04832-7.

Version 1.0

Programmierumgebung

Die vorliegenden Unterrichtsunterlagen wurden für die Programmierumgebung XLogo entwickelt. XLogo ist auf der Webseite `xlogo.tuxfamily.org` kostenlos verfügbar.

Damit die Logo-Programme aus den Unterlagen ausgeführt werden können, muss XLogo auf Englisch eingestellt werden.

Nutzungsrechte

ABZ

Das Ausbildungs- und Beratungszentrum für Informatikunterricht der ETH Zürich unterstützt Schulen und Lehrkräfte, die ihren Informatikunterricht entsprechend auf- oder ausbauen möchten, mit einem vielfältigen Angebot. Es reicht von individueller Beratung und Unterricht durch ETH-Professoren und das ABZ-Team direkt vor Ort in den Schulen über Ausbildungs- und Weiterbildungskurse für Lehrkräfte bis zu Unterrichtsmaterialien.

`www.abz.inf.ethz.ch`

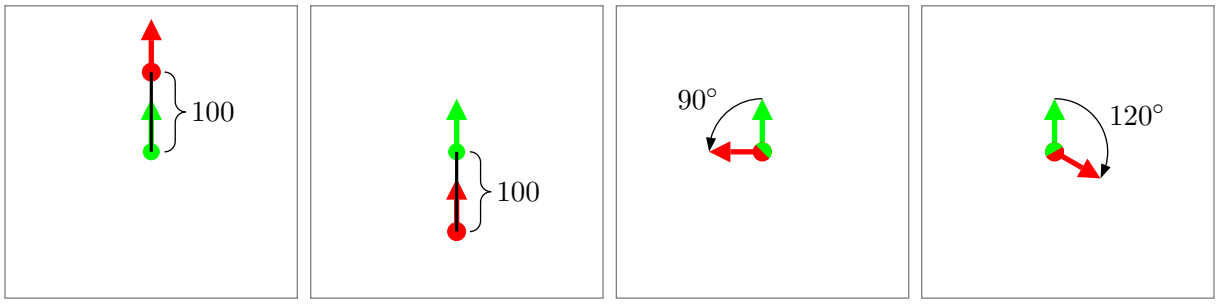
1 Was ist Programmieren?

Ein Computerbefehl ist eine Anweisung, die der Computer versteht und ausüben kann. Der Computer kennt eigentlich nur sehr wenige Befehle und alle komplizierten Tätigkeiten, die wir vom Computer vollbracht haben wollen, müssen wir aus den einfachen Computerbefehlen zusammensetzen. Diese Folge von Computerbefehlen nennen wir Programm. Programme zu schreiben ist nicht immer einfach. Es gibt Programme, die aus Millionen von Befehlen zusammengesetzt sind. Hierbei die Übersicht nicht zu verlieren, erfordert ein durchdachtes und sauberes Vorgehen, das wir in diesem Programmierkurs erlernen werden.

Jetzt kann's losgehen!

2 Grundbefehle

Mit den folgenden Befehlen kann die Schildkröte bewegt werden. Der grüne Punkt markiert jeweils die Startposition, der grüne Pfeil die Startrichtung. Für die Endposition und Endrichtung stehen der rote Punkt und der rote Pfeil.



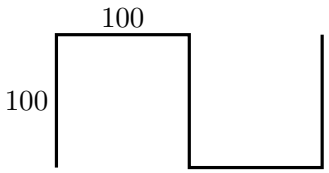
forward 100 | back 100 | left 90 | right 120

Damit man nicht so viel tippen muss, gibt es für diese Befehle Abkürzungen, die aus dem ersten und letzten Buchstabe bestehen: **fd**, **bk**, **lt** und **rt**.

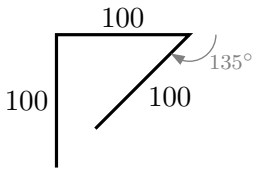
Um eine Figur zu löschen benutzt man den Befehl **cs**, was abgekürzt steht für **clearscreen**.

① Zeichne mit diesen Befehlen folgende Figuren.

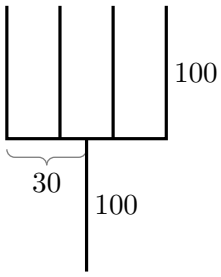
(a)



(b)

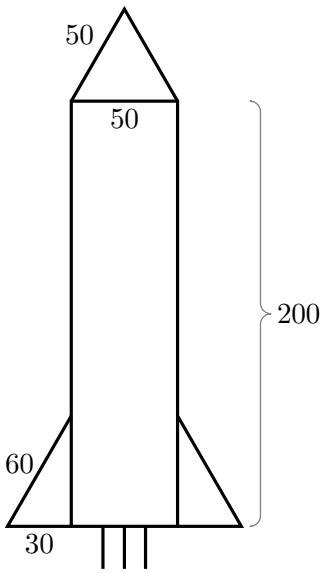


(c)



Für die Schnellen

② Zeichne folgende Rakete. Die Spitze ist ein gleichseitiges Dreieck mit Seitenlänge 50. Die Winkel in den Seitenruder-Dreiecken sind 30°, 60° und 90°. Die Länge und die Position der Flammen darfst du selber wählen.

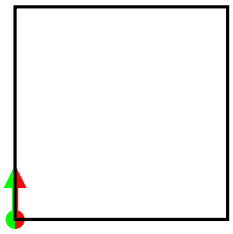


3 Wiederholungen

Oft trifft man beim Programmieren auf die Situation, dass gewisse Teile wiederholt werden müssen. Es wäre äusserst mühsam, wenn man die zugehörigen Befehle immer und immer wieder einzeln hinschreiben müsste. Daher gibt es einen eigenen Befehl für die Wiederholung: Mit **repeat 4 [...]** werden Anweisungen in der Klammer gleich 4 Mal ausgeführt.

So zeichnet zum Beispiel die folgende Zeile ein Quadrat der Seitenlänge 100:

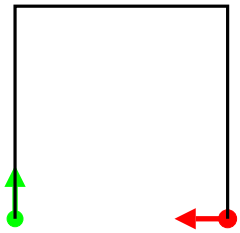
```
repeat 4 [fd 100 rt 90]
```



Natürlich kannst du die Zahl **4** abändern, wenn du mehr oder weniger Wiederholungen wünscht.

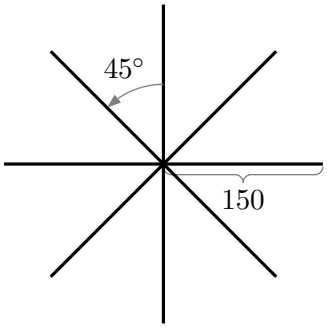
So zeichnet zum Beispiel die folgende Zeile ein nach unten offenes Quadrat der Seitenlänge 100:

```
repeat 3 [fd 100 rt 90]
```

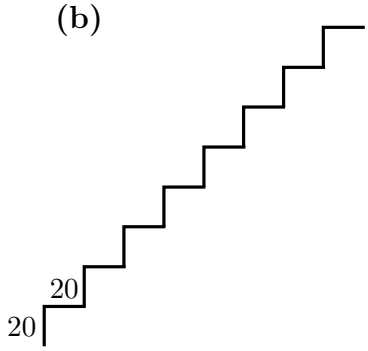


① Zeichne folgende Figuren. Benutze dabei den Befehl **repeat**:

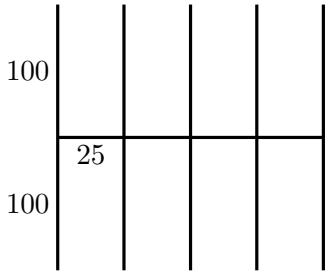
(a)



(b)



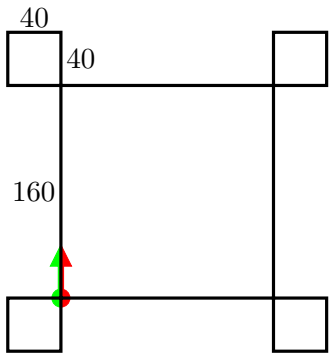
(c)



Ein **repeat** kann selbst wieder ein **repeat** enthalten. Man nennt diese **repeat *verschachtelt***. Betrachte dazu folgendes Beispiel

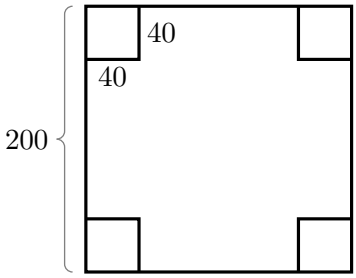
So zeichnet zum Beispiel die folgende Zeile ein Quadrat der Seitenlänge 160 mit angehängten Quadrätchen der Seitenlänge 40.

```
repeat 4 [fd 160 repeat 4 [fd 40 lt 90] rt 90]
```

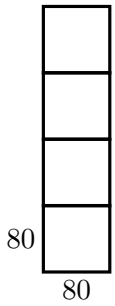


② Zeichne folgende Figuren. Benutze dabei verschachtelte **repeat**.

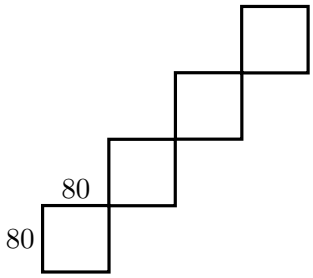
(a)



(b)



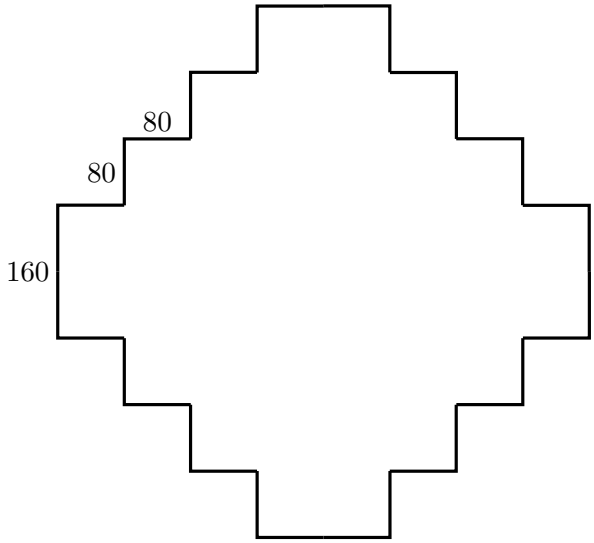
(c)



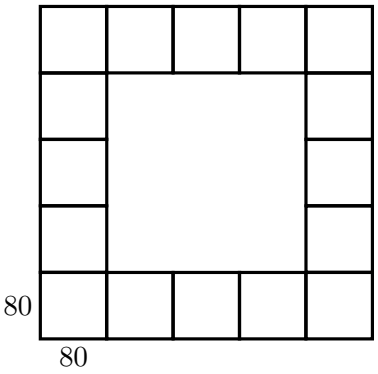
Für die Schnellen

③ Zeichne folgende Figuren.

(a)



(b)

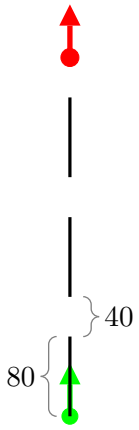


4 Der Wandermodus

Mit dem Befehl **penup**, abgekürzt **pu**, wird der Stift angehoben. Die Schildkröte befindet sich dann im ***Wandermodus***, sie zeichnet nichts mehr. Mit **pendown**, abgekürzt **pd**, wird der Stift wieder auf das Zeichenpapier runtergelassen. Betrachte folgendes Beispiel:

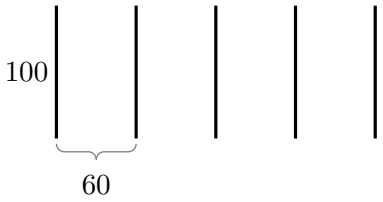
Dreimal wird der Stift angehoben und wieder abgesetzt.

repeat 3 [fd 80 pu fd 40 pd]

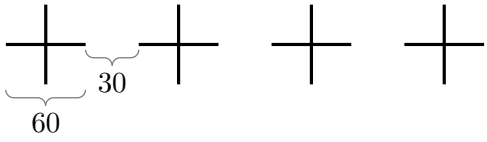


① Zeichne folgende Figuren.

(a)

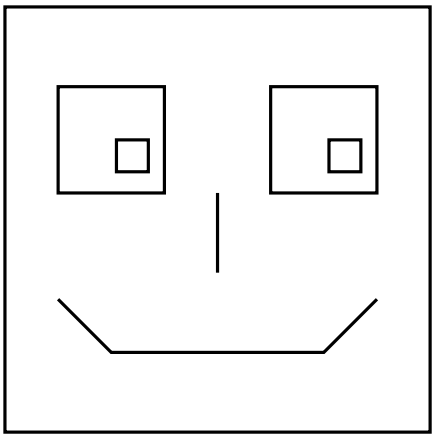


(b)



Für die Schnellen

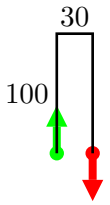
② Zeichne folgendes Gesicht. Die genauen Masse darfst du selber bestimmen.



5 Programme benennen

Figurenteile, die man häufiger braucht, kann man eigene Namen geben und dann verwenden. Dazu benutzt man die ***Befehle*** **to ... end**:

```
to lakula
  fd 100 rt 90
  fd 30 rt 90
  fd 100
end
```



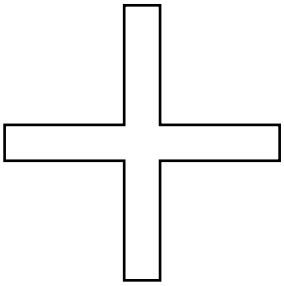
Einen solchen Figurenteil nennt man auch ***Programm***. Der seltsame Namen **lakula** steht für "lang-kurz-lang".

Wichtig ist es, sich zu überlegen, wo die Schildkröte zu Beginn und am Ende ist und wohin sie guckt. Dies haben wir im obigen Bild wie üblich durch Punkte und Pfeile angedeutet.

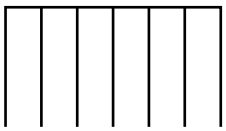
Die Befehlszeile **repeat 2 [lakula]** zeichnet somit die Figur:

① Benutze nun das Programm **lakula** und zeichne damit folgende Figuren.

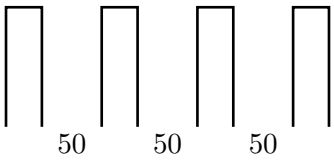
(a)



(b)



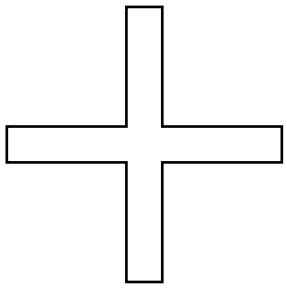
(c)



② Definiere nun ein eigenes Programm **kreuz**, das genau den Figurenteil aus ① **(a)** zeichnet:

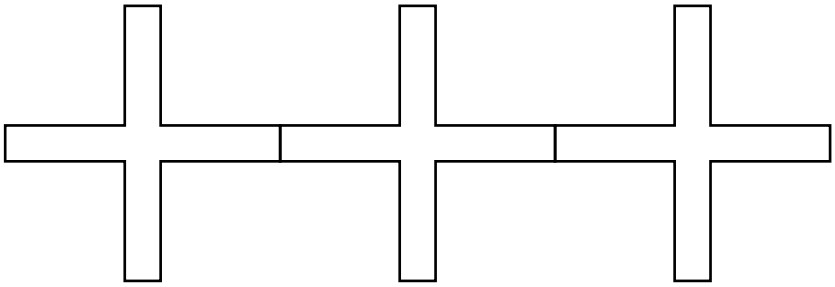
```
to kreuz
  ⋮
  ⋮
  ⋮
end
```

Hier kommen die Zeilen hin, mit denen du die Figur aus Aufgabe ① (a) gezeichnet hast.

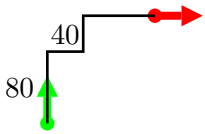


(a) Zeichne in das Bild rechts folgendes ein: Startposition (grüner Punkt), Startrichtung (grüner Pfeil), Endposition (roter Punkt), Endrichtung (roter Pfeil).

(b) Benutze nun das Programm **kreuz**, um folgende Figur zu zeichnen:

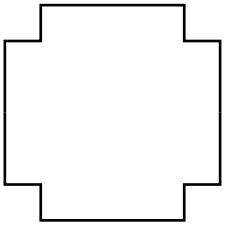


③ Definiere ein Programm **haken**, welches die Figur links zeichnet.

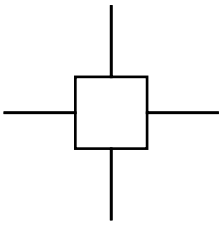


④ Benutze das Programm **haken**, um folgende Figuren zu zeichnen:

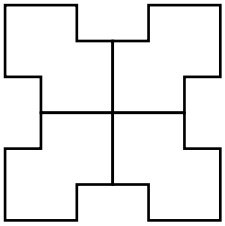
(a)



(b)



(c)

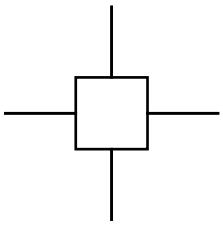


Für die Schnellen

⑤ Definiere nun ein eigenes Programm **stern**, das genau den Figurenteil aus ④ **(b)** zeichnet:

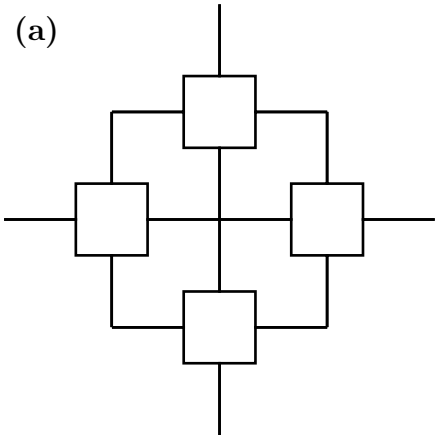
```
to stern
  ⋮
  ⋮
  ⋮
end
```

Hier kommen die Zeilen hin, mit denen du die Figur aus Aufgabe ④ (b) gezeichnet hast.

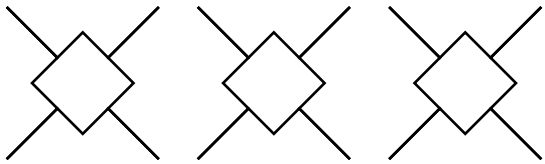


⑥ Benutze nun diesen **stern**, um folgende Figuren zu zeichnen:

(a)



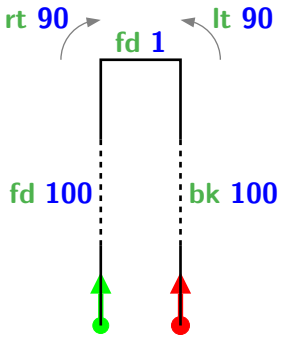
(b)



6 Flächen füllen

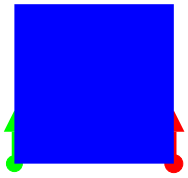
Das Programm **fett100** zeichnet eine dicke Linie. Das Bild in der Mitte zeigt das Resultat: eine fette Linie. Rechts daneben wird die Abfolge schematisch dargestellt.

```
to fett100
  fd 100
  rt 90
  fd 1
  lt 90
  bk 100
end
```

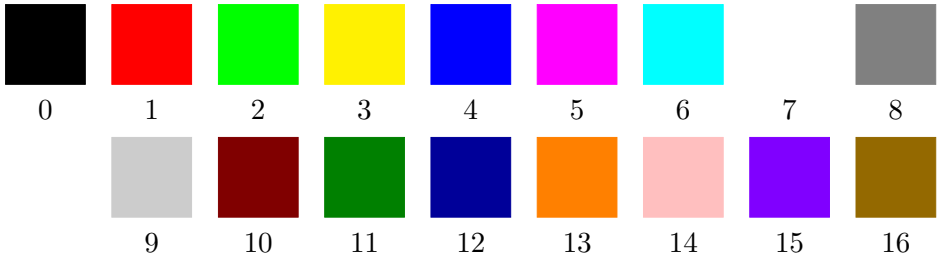


Damit können wir nun ein ausgefülltes Quadrat zeichnen:

```
setpc 4 repeat 100 [fett100]
```



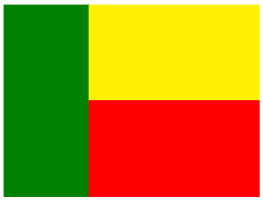
Der Befehl **setpc** ist eine Abkürzung für **setpencolor**, die Wahl der Stiftfarbe. Die Farben sind durchnummeriert:



① Zeichne zwei der folgenden Fahnen (sie sollen 360 breit und 240 hoch sein):

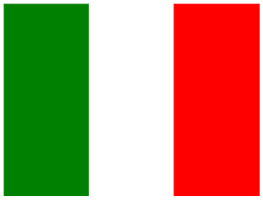
Armenien

Belgien



Benin

Deutschland



Italien

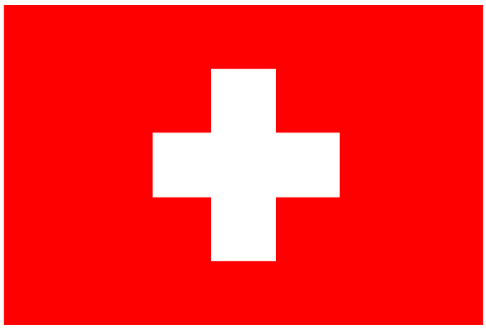
Litauen

Mauritius

Österreich

Für die Schnellen

(2) Zeichne eine der folgenden Flaggen:



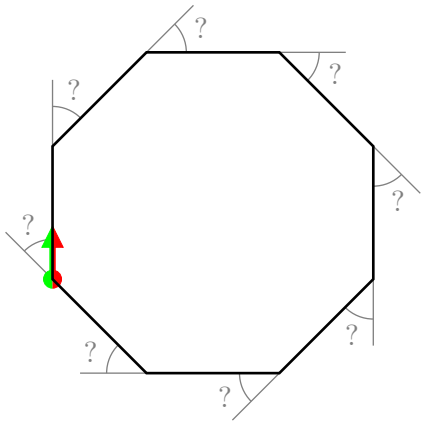
Schweiz

Griechenland

7 Regelmässige Vielecke und Kreisbögen

Bei einem ***regelmässigen*** Vieleck sind alle Seiten gleich lang und alle Innenwinkel gleich gross. Wenn die Schildkröte ein solches Vieleck zeichnen soll, so endet sie am Ende wieder am selben Ort wie sie gestartet ist, und sie hat sich ***genau einmal um*** 360° ***gedreht***.

```
repeat 8 [fd 100 rt ?]
```



Bei einem 8-Eck, muss der volle Winkel 360° daher in ? = 360°/8 gleiche Winkel aufgeteilt werden. Nun ist 360°/8 = 45°, aber das muss man nicht wissen, denn der Computer kann dies selbst errechnen:

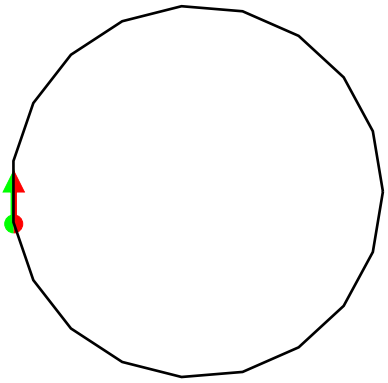
```
repeat 8 [fd 100 rt 360/8]
```

① Zeichne die folgenden regelmässige Vielecke:

- **(a)** Ein 5-Eck mit der Seitenlänge 180,
- **(b)** Ein 12-Eck mit der Seitenlänge 50,
- **(c)** Ein 7-Eck mit der Seitenlänge 100.

Ein regelmässiges Vieleck mit sehr vielen Seiten ähnelt einem Kreis. Schon bei einem 20-Eck kann man kaum noch die Ecken erkennen:

```
repeat 20 [fd 20 rt 360/20]
```



Wir werden daher Kreise als regelmässige Vielecke mit sehr vielen (typischerweise 360) Seiten zeichnen.

② Teste die folgenden Programme:

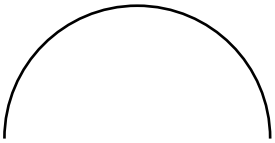
(a) **repeat 360 [fd 1 rt 1]**

(b) **repeat 180 [fd 3 rt 2]**

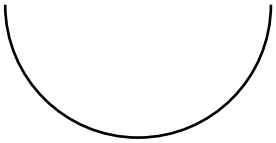
(c) **repeat 360 [fd 2.5 rt 1]** (2.5 bedeutet 2 und einen halben Schritt)

③ Zeichne folgende Halbkreise. Die Grösse darfst Du selber bestimmen.

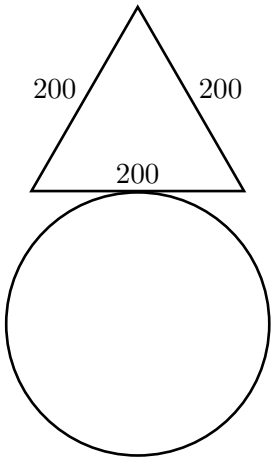
(a)



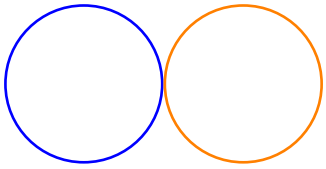
(a)



④ Zeichne folgenden "Clown". Die Grösse des Kreises darfst du selber wählen.

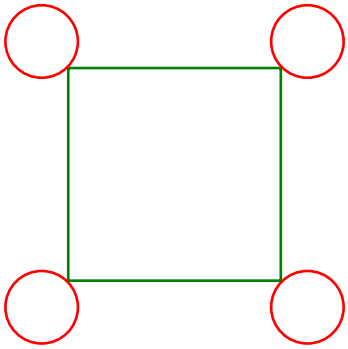


⑤ Zeichne folgende Figur.



Für die Schnellen

⑥ Zeichne folgende Figur.



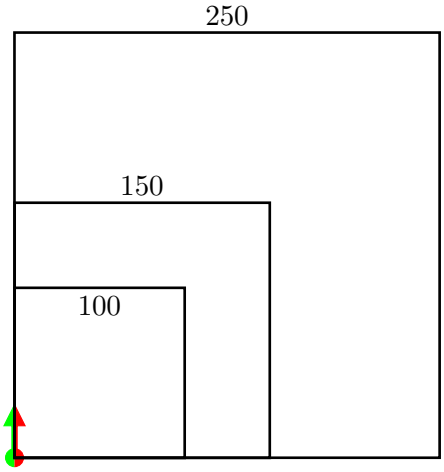
8 Programme mit Variablen

Betrachte folgendes Programm, in dem die Variable **:lang** vorkommt:

```
to quadrat :lang
  repeat 4 [
    fd :lang
    rt 90
  ] end
```

Dieses kann nun verwendet werden:

```
quadrat 100
quadrat 150
quadrat 250
```



Beim ersten Aufruf, also bei **quadrat 100**, wird die Variable **:lang** den Wert 100 annehmen. Beim zweiten Aufruf wird dieselbe Variable den Wert 150 und beim dritten den Wert 250 annehmen.

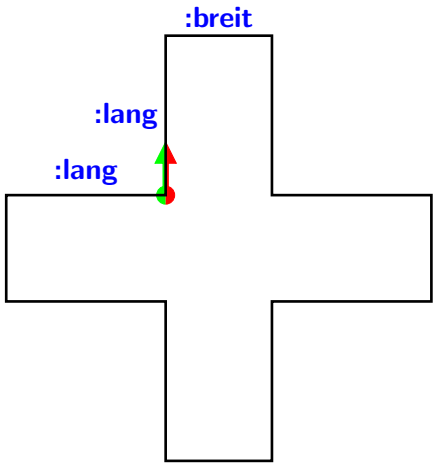
Ein Programm kann auch mehrere Variablen verwenden. Zum Beispiel

```
to vieleck :lang :anz
  repeat :anz [
    fd :lang
    rt 360/:anz
  ]
end
```

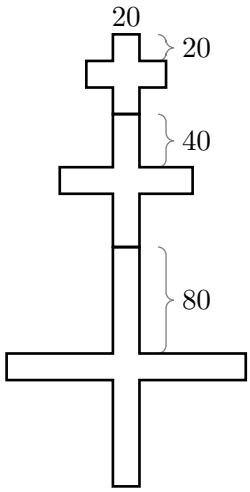
Damit kann man regelmässige Vielecke beliebiger Grösse und beliebig vielen Ecken zeichnen.

① Schreibe ein Programm **kreuzvar** mit zwei Argumenten **:lang** und **:breit**, welches ein Kreuz mit diesen Abmessungen zeichnet, siehe die Figur rechts.

```
to kreuzvar :lang :breit
  ⋮
  ⋮
end
```

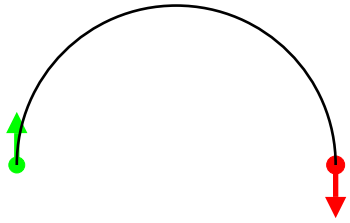


② Zeichne nun mit dem Programm **kreuzvar** folgende Figur:



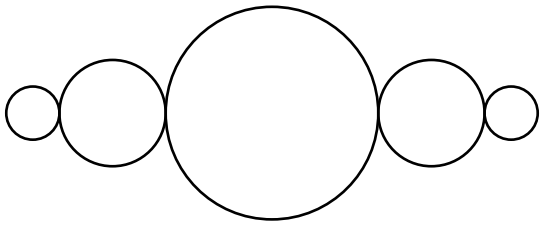
③ Schreibe ein Programm **bogen** mit der Variablen **:gr**, das einen Halbkreis zeichnet, welcher mit 180 Strecken der Länge **:gr** gezeichnet wird.
Beispiel:

bogen 3



④ Zeichne die folgende Figur.

Verwende dazu Bögen der Grösse 0.5, 1 und 2.



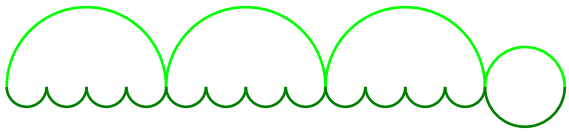
⑤ Schreibe ein Programm **bogenreihe** mit den Variablen **:gr** und **:anz**, welches **:anz** Halbkreisbögen zeichnet. Verwende das Programm **bogen** aus der Aufgabe ③.

Beispiel:

bogenreihe 5 1

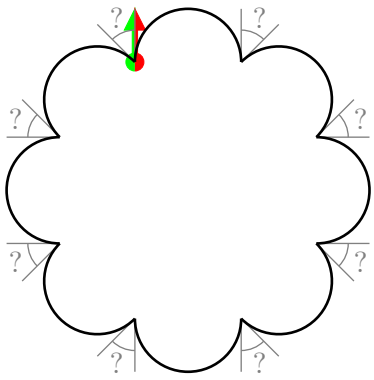
⑥ Verwende das Programm **bogenreihe** und zeichne folgende Raupe.

Verwende dazu Bögen der Grösse 2, 1 und 0.5.



⑦ Die nebenstehende Figur ist ganz ähnlich wie die, welche das Programm **bogenreihe** zeichnet, nur dass zwischen zwei Bögen noch extra ein Winkel dazugeschaltet wurde.

Schreibe ein Programm **wolke** mit den Variablen **:gr** und **:anz**, welche eine solche Wolke mit **:anz** Ausbuchtungen der Grösse **:gr** zeichnet.

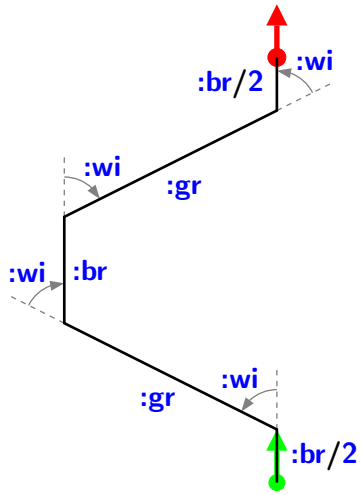


⑧ Nun soll eine Zahnradstange gezeichnet werden. Schreibe dazu ein Programm

zahnradstange

mit den Variablen **:gr**, **:br**, **:wi**, **:anz**. Die ersten drei Variablen bestimmen die Form eines Zahns, siehe die Abbildung rechts.

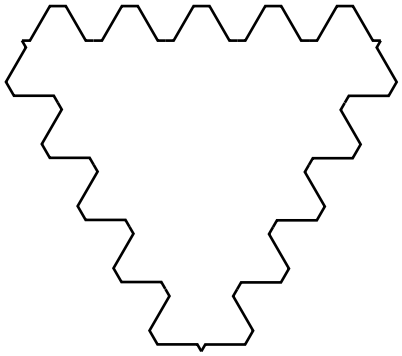
Die Variable **:anz** gibt an, wieviele Zähne die Stange hat, siehe die Beispiele unten.



rt 90 zahnradstange 30 20 90 8

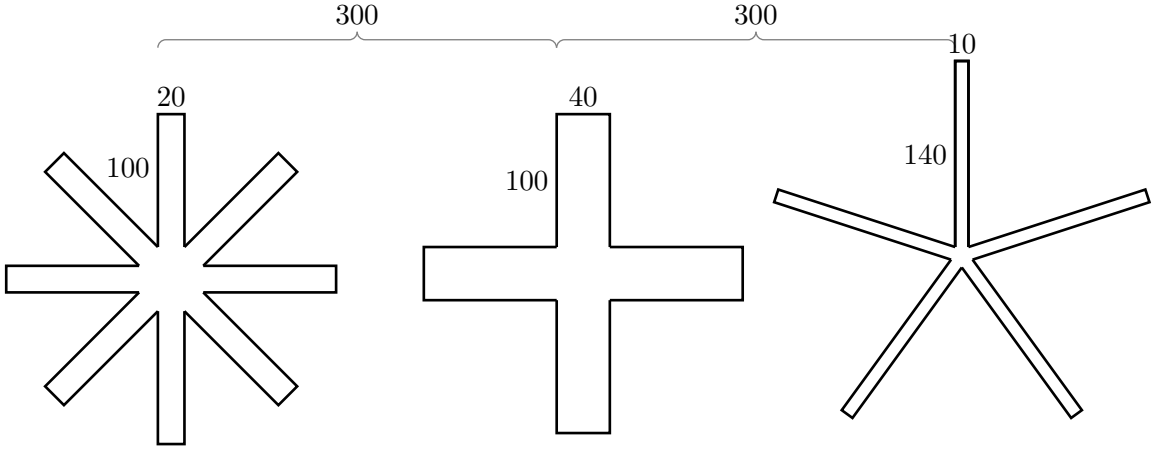
rt 90 zahnradstange 40 20 70 6

⑨ Zeichne mit dem Programm **zahnradstange** folgendes gezahnte Dreieck, mit **:gr** = 50, **:br** = 20 und **:wi** = 60°.



Für die Schnellen

⑩ Schreibe ein Programm **balkenstern :gr :br :anz**, das einen Stern mit **:anz** Balken der Länge **:gr** und Breite **:br** zeichnet. und erstelle damit folgendes Bild.



9 Bewegte Bilder

Eine Abfolge von Bildern, bei denen sich ein Objekt von Bild zu Bild nur leicht verändert, wird vom menschlichen Auge als Bewegung erfasst. Eine solche Abfolge nennt man ***Animation***. Wir werden dies hier ausprobieren.

Der Befehl **penerase**, kurz **pe**, macht aus dem Stift ein "Radiergummi". Mit dem Befehl **penpoint**, kurz **ppt**, kann wieder in den normalen Zeichenmodus zurückgewechselt werden.

Mit dem Befehl **wait 4** kann die Ausführung verzögert werden. Die Zahl **4** steht für die Wartezeit. Dies ist wichtig, damit das Bild für eine kurze Zeit stehen bleibt und nicht sofort austradiert wird, bevor unser Auge es erfassen kann.

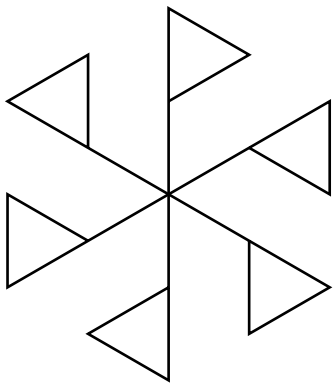
① Studiere folgendes Programm. Schreibe es ab und probiere es aus.

```
to quadlauf
  repeat 120 [
    quadrat 100
    wait 4
    pe
    quadrat 100
    rt 90
    fd 4
    lt 90
    ppt
  ]
end
```

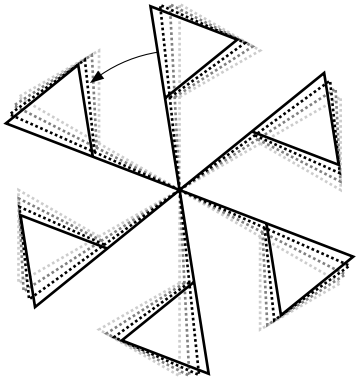
② Modifiziere das Programm aus ① so, dass sich das Quadrat doppelt so schnell und doppelt so weit bewegt.

③ Das Programm **windrad** zeichnet ein Windrad mit sechs dreieckigen Blättern.

```
to windrad
  repeat 6 [
    fd 50
    vieleck 50 3
    bk 50
    rt 60
  ]
end
```

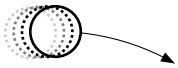


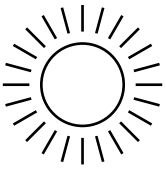
Tippe dieses Programm ab und erzeuge damit eine Animation, bei der sich das Windrad fünf Mal im Gegenuhrzeigersinn dreht.



Für die Schnellen

④ Zeichne einen Planeten, der eine Sonne umkreist. Die Sonne soll still stehen, nur der Planet soll kreisen.





10 Zufallsbilder

Der Computer kann zufällige Zahlen erzeugen: der Befehl

```
random 50
```

liefert eine zufällige Zahl zwischen 0 und 50.

Möchte man eine zufällige Zahl zwischen 10 und 30 so tippt man

```
10 + random 20
```

(1) Notiere in der Tabelle einige nützliche Verwendungen von Zufallszahlen:

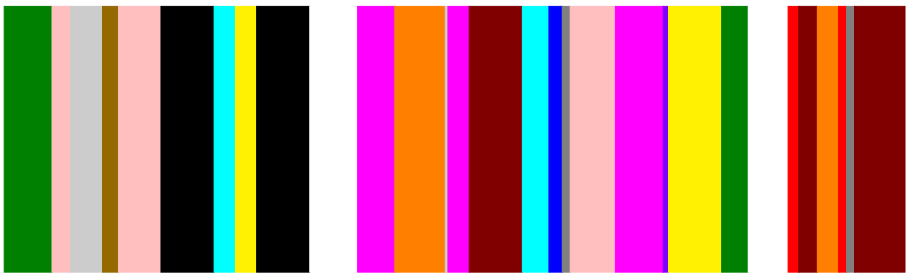
`setpc random 16`	Setzt die Stiftfarbe zufällig.
	Rückt um eine zufällige Zahl zwischen 0 und 100 vor.
	Dreht um einen beliebigen Winkel.

(2) Schreibe ein Programm **zufallsrechteck**, das ein ausgefülltes Rechteck einer zufälligen Breite (von 0 bis 20) der Höhe 100 in einer zufälligen Farbe zeichnet.

(3) Schreibe ein Programm **zufallsbild**, das ein Bild mit einer variablen Anzahl Streifen von variablen Breite und der Höhe 100 zeichnet.

Genauer:

Gezeichnet werden sollen **:anz** vertikale Streifen gleicher Höhe aber mit einer variablen Breite, die maximal **:br** sein kann und variabler Farbe.



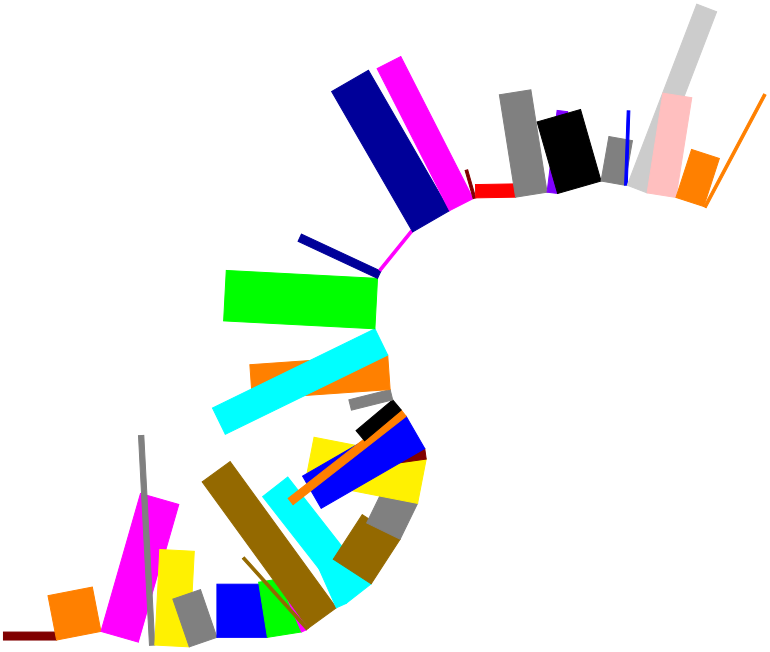
zufallsbild 30 20 (30 Streifen mit maximaler Breite 20)

Lasse das Programm mehrfach laufen und erzeuge verschiedene Zufallsbilder damit.

(4) Ändere das Programm nun so ab, dass auch die Höhe der Rechtecke zufällig zwischen 0 und **:ho** variieren kann und nach jedem Rechteck zufällig um einen Winkel zwischen $-30°$ und $+30°$ nach rechts gedreht wird (bei einem negativen Winkel wird dann nach links gedreht). Dies erfolgt mit

```
rt (random 60) - 30
```

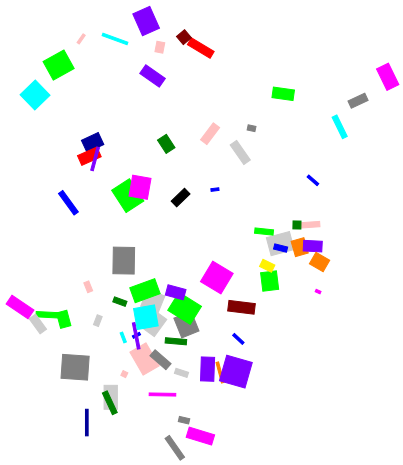
Ein mögliches Bild ist:



zufallsbildzwei 40 20 80
(40 Balken der maximalen Breite 20 und maximalen Höhe 80)

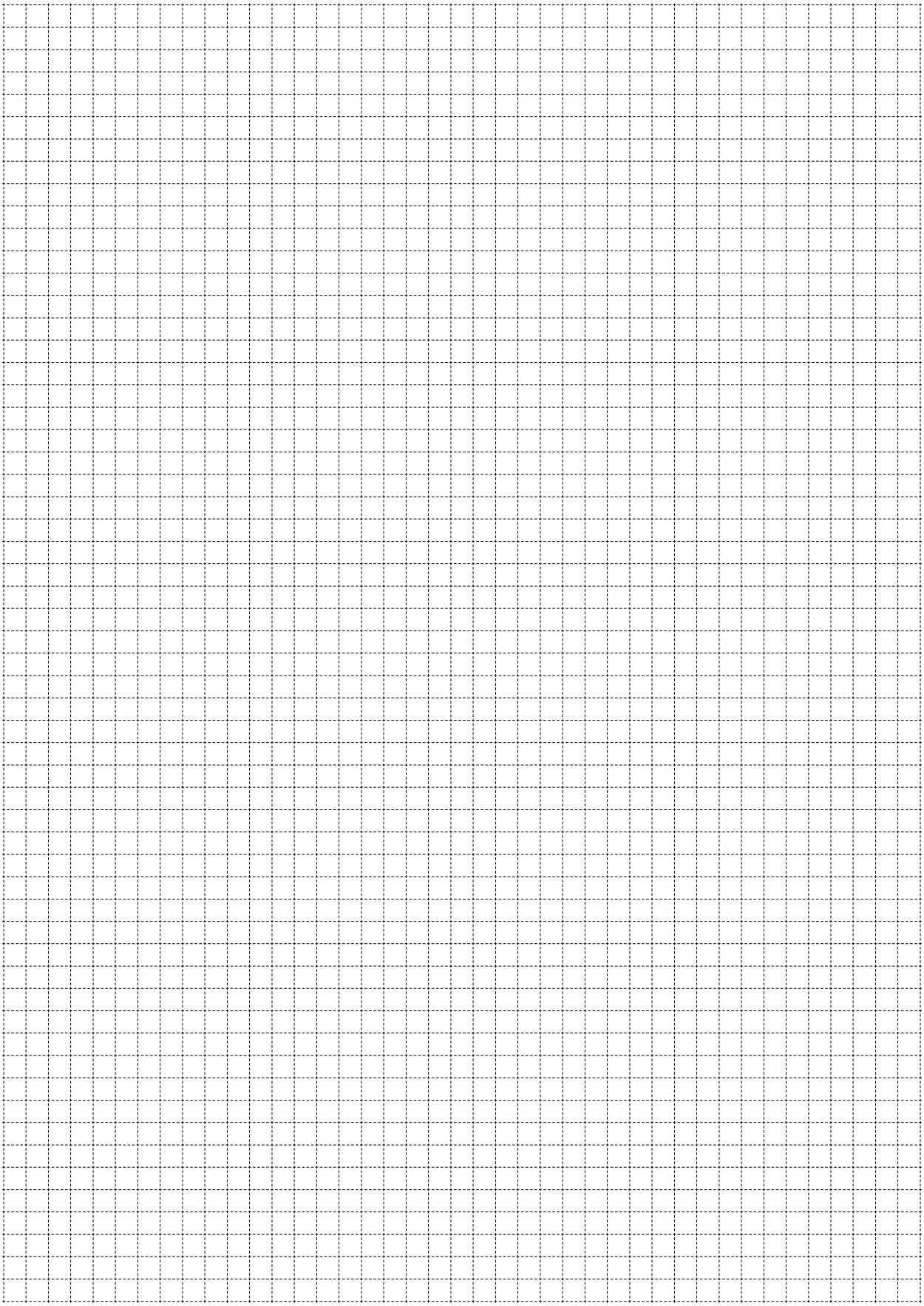
Für die Schnellen

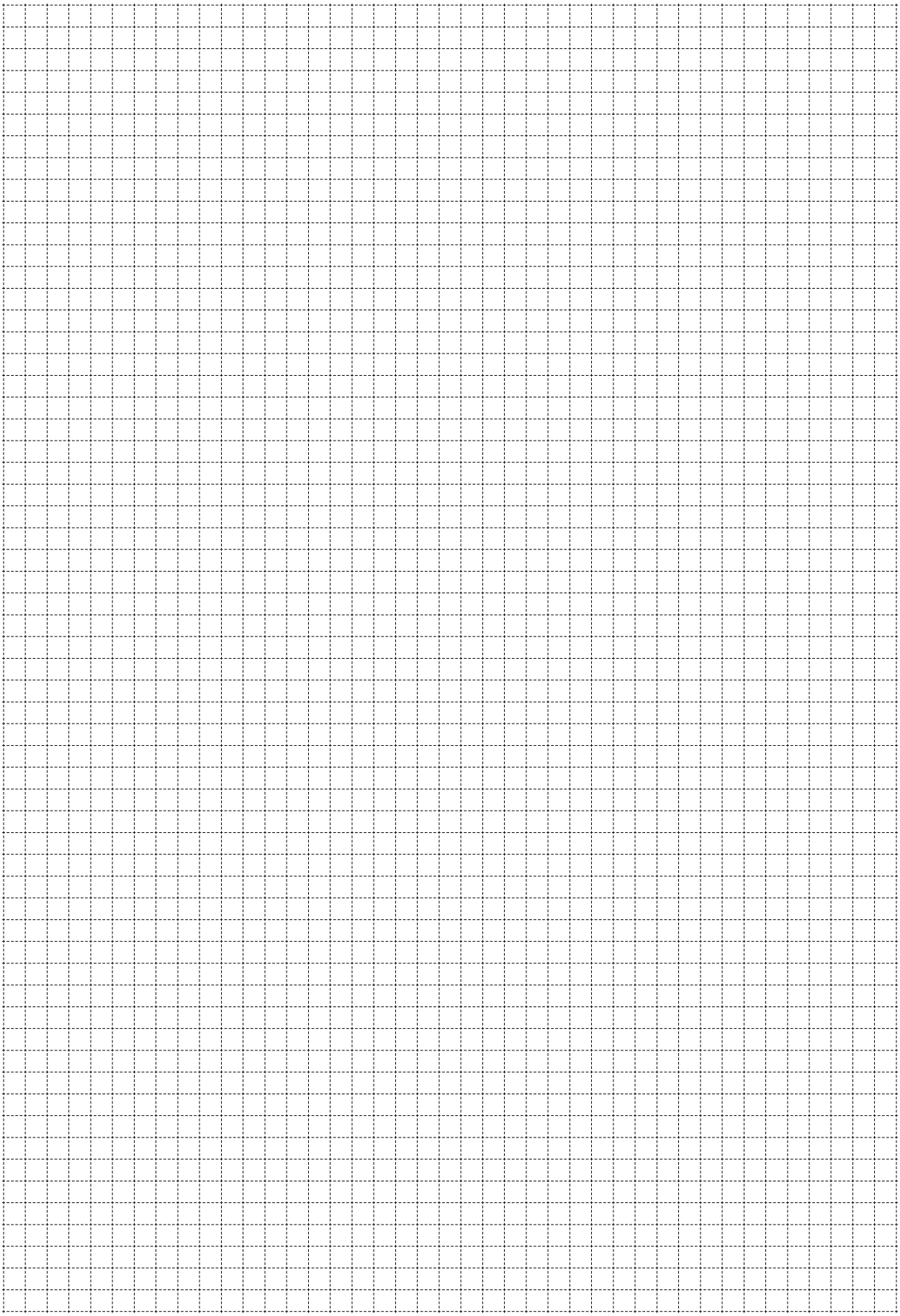
Ⓢ Erzeuge ein Zufallsbild, bei dem **:anz** viele Rechtecke gezeichnet werden, die maximal **:gr** breit und maximal **:gr** hoch sein können. Zwichen den verschiedenen Rechtecken dreht sich die Schildkröte um einen zufälligen Winkel und kriecht dann um eine zufällige Distanz vorwärts. Die Distanz kann maximal **:dist** sein.



zufallsbilddrei 80 10 30
(80 Rechtecke der maximalen Breite und Höhe 10, der Abstand von Rechtecke zu Rechteck ist maximal 30)

Meine Notizen:





Befehlsübersicht

fd 100	100 Schritte vorwärts gehen
bk 50	50 Schritte rückwärts gehen
cs	alles löschen und neu beginnen
rt 90	90 Grad nach rechts drehen
lt 90	90 Grad nach links drehen
repeat 4 [...]	das Programm in **[...]** wird viermal wiederholt
pu	die Schildkröte wechselt in den Wandermodus
pd	die Schildkröte wechselt zurück in den Stiftmodus
setpc 3	wechselt die Stiftfarbe auf die Farbe 3
to Name	erstellt ein Programm mit einem Namen
to Name :variable	erstellt ein Programm mit einem Namen und einer Variablen
end	alle Programme mit einem Namen enden mit diesem Befehl
pe	die Schildkröte wechselt in den Radiergummimodus
ppt	die Schildkröte wechselt zurück in den Stiftmodus
wait 5	die Schildkröte wartet 5 Zeiteinheiten
random 50	liefert eine zufällige Zahl zwischen 0 und 50

Programmieren mit LOGO

Informationstechnologie und Ausbildung
ETH Zürich, CAB F 15.1
Universitätstrasse 6
CH-8092 Zürich

www.ite.ethz.ch
www.abz.inf.ethz.ch