Lösungen zu

Einführung in die Programmierung mit LOGO

Juraj Hromkovič

4. Auflage

Kapitel 1

Aufgabe 1.1 Die Schildkröte geht mit diesen drei Befehlen jeweils vorwärts und zeichnet eine zusammenhängende, gerade Linie. Das gleiche Resultat würde sich auch ergeben, wenn du direkt `forward 170` schreibst, denn $100 + 50 + 20 = 170$.

Aufgabe 1.2 `fd 50` zeichnet eine Linie mit Länge 50, genau wie der Befehl `forward 50`.

Aufgabe 1.3 `cs` löscht den Bildschirm und setzt die Schildkröte auf die Startposition mit Blick nach oben.

Aufgabe 1.4 Die Schildkröte zeichnet eine gerade, senkrechte Linie mit einer Gesamtlänge von 300. Von der Startposition aus gesehen befindet sich eine Linie der Länge 100 *vor* und eine Linie der Länge 200 *hinter* der Schildkröte. Also ist die Linie insgesamt 300 lang.

Aufgabe 1.6 Der erste Befehl `rt 360` bewirkt nichts, da sich die Schildkröte genau einmal in einem ganzen Kreis um sich selber dreht und somit wieder in die gleiche Richtung schaut wie vor Ausführung des Befehls. Danach wird der zweite Befehl `rt 180` ausgeführt. Da sie anfangs nach oben geschaut hat und nun mit einer Drehung um 180° genau eine halbe Drehung vornimmt, schaut sie nun nach unten. Mit dem dritten Befehl `rt 90` dreht sich die Schildkröte aus *ihrer* Perspektive (nicht aus deiner) eine Vierteldrehung nach rechts, und schaut nun nach links. Mit dem vierten und letzten Befehl `rt 180` dreht die Schildkröte wieder eine halbe Drehung. Da sie vorher nach links schaute, schaut sie jetzt abschliessend nach rechts.

Die vier Befehle lassen sich auch zusammenfassen. Eine Vierteldrehung nach rechts, so dass die Schildkröte, die anfangs nach oben schaute, nun nach rechts schaut, lässt sich einfach mit `rt 90` erreichen.

Aufgabe 1.7 `lt 180` ist gleichbedeutend mit `rt 180`. Mit beiden Befehlen nimmt die Schildkröte genau eine halbe Drehung vor. Beim ersten Befehlt links herum, beim zweiten rechts herum. Auf das Resultat hat die Drehrichtung bei einer halben Drehung keinen Einfluss.

`lt 270` ist gleichbedeutend mit `rt 90`. Du hast gelernt, dass eine *ganze* Drehung einen Winkel von 360° bedeutet. Von der Startposition aus kannst du also entweder 270° nach links drehen, um die Schildkröte auf dem Bildschirm nach rechts schauen zu lassen, oder du kannst um 90° nach rechts drehen, um das gleiche Resultat zu erhalten.

Aufgabe 1.10 Durch die `fd`-Befehle wird jeweils ein Teilstrich der Figur gezeichnet, die `rt 90`-Befehle bewirken jeweils eine Drehung nach rechts. Somit entsteht eine Art spiralförmige Figur.

Aufgabe 1.12 Wenn du alle Befehle `rt 90` ersetzt durch `lt 270`, dann bekommst du genau die gleiche Figur.

```
fd 100 lt 270 fd 150 lt 270 fd 50 lt 90 fd 150 lt 270 fd 50
```

Aufgabe 1.13 Der `bk 100`-Befehl lässt sich ersetzen durch eine halbe Drehung (180°) der Schildkröte, dann einem Vorwärtslaufen um 100 (`fd 100`), und abschliessend wieder einer Drehung um 180°. Letzteres ist notwendig, damit die Schildkröte wieder in die gleiche Richtung schaut wie im ursprünglichen Programm. Durch dieses einfache Ersetzen müssen wir die anderen Befehle nicht anpassen.

Die beiden Befehle `lt 90` ersetzen wir einfach durch `rt 270`.

```
fd 120 rt 90 fd 120 rt 270
rt 180 fd 100 rt 180 rt 90
bk 100 rt 270 fd 80
```

Aufgabe 1.15

```
pe rt 180 fd 50 lt 90 fd 150 rt 90
fd 50 lt 90 fd 150 lt 90 fd 100
```

Aufgabe 1.16

```
rt 90
fd 50 pe fd 50 penpaint
fd 50 pe fd 50 penpaint
fd 50 pe fd 50 penpaint
fd 50
```

Aufgabe 1.17 Es entstehen zwei parallele senkrechte Linien mit der Länge 50.

Aufgabe 1.18

```
pe bk 100 rt 90 bk 100
penpaint rt 90 fd 100 rt 90 fd 100
```

Kontrollaufgabe 1 Die Befehlsfolge `rt 180 lt 90` lässt sich zusammenfassen zu `rt 90`. Der letzte Befehl `lt 360` ist überflüssig.

```
fd 100 rt 90 fd 50 rt 90 fd 100 rt 90 fd 50
```

Kontrollaufgabe 2

```
fd 50 rt 90 fd 50 lt 90
fd 50 rt 90 fd 50 lt 90
fd 50 rt 90 fd 50 rt 90
fd 50 lt 90 fd 50 rt 90
fd 50 lt 90 fd 50 rt 90
fd 50
```

Der Befehl `lt 90` lässt sich ersetzen durch `rt 270`, welcher sich wiederum aus drei aneinandergereihten `rt 90 rt 90 rt 90` zusammensetzen lässt. Somit ist es möglich, die Zeichnung nur mit den beiden Befehlen `fd 50` und `rt 90` zu erstellen.

Der Befehl `fd 50` lässt sich aneinanderreihen aus `fd 10 fd 10 fd 10 fd 10 fd 10`, um zum gleichen Resultat zu gelangen. Also ist es auch möglich, nur mit Hilfe der beiden Befehle `fd 10` und `rt 90` die Schildkröte die Zeichnung erstellen zu lassen.

Kontrollaufgabe 3

```
lt 90 fd 25 rt 90 fd 60 lt 90 fd 10 bk 10 rt 90 bk 10
rt 90 fd 50 lt 90 fd 10 rt 90 fd 10 bk 10 rt 90 fd 60
rt 90 fd 25 lt 90 fd 50 lt 90 fd 25 rt 90 fd 25 lt 90
fd 50 lt 90 fd 5 bk 10 fd 5 lt 90 fd 50 lt 90 fd 75
rt 90 fd 10 lt 90 fd 50 lt 90 fd 5 bk 10 fd 5 lt 90
fd 50 lt 90 fd 30 lt 90 fd 50 lt 90 fd 5 bk 10 fd 5
lt 90 fd 50 lt 90 fd 10 rt 90 fd 75 lt 90 fd 50 lt 90
fd 5 bk 10 fd 5 lt 90 fd 50 lt 90 fd 25 rt 90 fd 25
```

Kontrollaufgabe 4

```
fd 50 rt 90 fd 50 rt 90 fd 50 lt 90
fd 50 rt 90 fd 50 rt 90 fd 50 lt 90
fd 50 rt 90 fd 50 rt 90 fd 50 lt 90
fd 50 rt 90 fd 50 rt 90 fd 100 rt 90
fd 50 rt 90 fd 50 rt 90 fd 50
```

Kontrollaufgabe 5

```
pe
rt 180 fd 50 lt 90
fd 50 rt 90 fd 50 lt 90
fd 50 rt 90 fd 50 lt 90
fd 50 lt 90 fd 50 rt 90
fd 50 lt 90 fd 50 rt 90
fd 50 lt 90 fd 50
```

Kontrollaufgabe 6

```
pe
fd 50 rt 90 fd 50 rt 90 fd 50 lt 90
fd 50 rt 90 fd 50 rt 90 fd 50 lt 90
fd 50 rt 90 fd 50 rt 90 fd 50 lt 90
fd 50 rt 90 fd 50 rt 90 fd 100 rt 90
fd 50 rt 90 fd 50 rt 90 fd 50
```

Kontrollaufgabe 7

```
pe rt 180 fd 50 penpaint
fd 50 lt 90 fd 50 lt 90 fd 50 rt 90
fd 150 rt 90 fd 50 rt 90 fd 50 rt 90 fd 50
```

Kontrollaufgabe 8

```
fd 100 pe rt 90 fd 100
penpaint rt 90 fd 100 pe lt 90 fd 100
penpaint lt 90 fd 100 pe rt 90 fd 100
penpaint rt 90 fd 100
```

Kontrollaufgabe 9

```
pe rt 180 fd 50
penpaint lt 90 fd 50 pe rt 90 fd 50
penpaint lt 90 fd 50 pe rt 90 fd 50
penpaint lt 90 fd 50 pe lt 90 fd 50
penpaint rt 90 fd 50 pe lt 90 fd 50
penpaint rt 90 fd 50 pe lt 90 fd 50
```

Kontrollaufgabe 10

```
fd 150 pe rt 90 fd 25
penpaint rt 90 fd 25 lt 90 fd 25 lt 90
fd 25 lt 90fd 25 lt 90 fd 25 pe rt 90 fd 25
penpaint rt 90 fd 75 lt 90 fd 25 bk 25 lt 90
fd 25 lt 90 fd 200 lt 90 fd 25 rt 90
fd 25 bk 25 rt 90 fd 75 pe rt 90 fd 25
penpaint rt 90 fd 25 lt 90 fd 25 lt 90
fd 25 lt 90 fd 25 lt 90 fd 25 pe rt 90 fd 25
penpaint rt 90 fd 50 pe rt 90 fd 100
penpaint lt 90 bk 25 fd 50 bk 25 pe lt 90 fd 100
penpaint rt 90 fd 100 rt 90 fd 200 rt 90 fd 25 pe rt 90 fd 50
penpaint fd 100 lt 90 fd 25 lt 90 fd 100 lt 90 fd 25
```

Kapitel 2

Aufgabe 2.1

```
repeat 4 [ fd 200 rt 90 ]
```

Aufgabe 2.2 Das Programm zeichnet ein Rechteck mit Breite 200 und Höhe 100. Da sich die Befehlsfolge `fd 100 rt 90 fd 200 rt 90` zweimal wiederholt, können wir auch mit Hilfe des `repeat`-Befehls

```
repeat 2 [ fd 100 rt 90 fd 200 rt 90 ]
```

schreiben.

Aufgabe 2.3 Das Programm zeichnet ein Sechseck.

```
repeat 6 [ fd 50 rt 60 ]
```

Aufgabe 2.4

```
repeat 3 [ fd 200 rt 120 ]
```

Aufgabe 2.5

(a)
```
repeat 10 [fd 20 rt 90 fd 20 lt 90 ]
```

(b)
```
repeat 5 [fd 50 rt 90 fd 50 lt 90 ]
```

(c)
```
lt 90 repeat 20 [fd 10 lt 90 fd 10 rt 90 ]
```

Aufgabe 2.6

```
repeat 4 [ fd 30 rt 90 fd 30 lt 90 ] rt 90
repeat 4 [ fd 30 rt 90 fd 30 lt 90 ] fd 60 lt 90
repeat 4 [ fd 30 rt 90 fd 30 lt 90 ] rt 90
repeat 4 [ fd 30 rt 90 fd 30 lt 90 ]
```

Aufgabe 2.7 Das Programm zeichnet vier Quadrate der Größe 100, welche wiederum zu einem grossen Quadrat der Größe 200 zusammengefügt sind.

Da auch der folgende Codeblock `repeat 4 [... ]rt 90` viermal hintereinander ausgeführt wird, können wir einmal versuchen, auch diesen Block in einen `repeat`-Befehl zu verpacken:

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

Aufgabe 2.8

(a)
```
repeat 8 [fd 150 bk 150 lt 45 ]
```

(b) In diesem Fall muss auch der Drehwinkel des `lt`-Befehls auf 22.5° angepasst werden:

```
repeat 16 [ fd 100 bk 100 lt 22.5 ]
```

Aufgabe 2.9

(a)
```
repeat 4 [repeat 4 [fd 30 rt 90 fd 30 lt 90 ]rt 90
          repeat 4 [fd 30 rt 90 fd 30 lt 90 ]fd 60 lt 90 ]
pe rt 90 bk 60
```

(b)
```
repeat 4 [repeat 2 [repeat 4 [fd 30 rt 90 fd 30 lt 90 ]rt 90 ]
          lt 90 fd 60 lt 90 ]
pe rt 90 bk 60
```

Aufgabe 2.10

```
repeat 10 [ repeat 2 [ fd 20 rt 90 ] repeat 2 [ fd 20 lt 90 ] ]
```

Aufgabe 2.12

Erste Lösung

Zuerst zeichnen wir ein Quadrat mit dem bekannten Befehl `repeat 4 [fd 20 lt 90 ]`. Um zur Startposition des nächsten, also des rechten, Quadrates zu gelangen, muss die Befehlsfolge `lt 90 fd 20 rt 90` verwendet werden. Damit sieht das gesamte Programm wie folgt aus:

```
repeat 10 [ repeat 4 [ fd 20 lt 90 ] lt 90 fd 20 rt 90 ]
```

Zweite Lösung
Zuerst zeichnen wir den Umfang mit `repeat 2 [fd 20 lt 90 fd 200 lt 90 ]`. Im zweiten Schritt folgen die einzelnen Zwischenstriche. Mit `lt 90 fd 20 rt 90` kommt die Schildkröte an die passende Position. Mit `fd 20 bk 20` wird die eigentliche Linie gezeichnet. Das Zeichnen der Zwischenlinien muss neun mal wiederholt werden. Also sieht das fertige Programm so aus:

```
repeat 2 [ fd 20 lt 90 fd 200 lt 90 ]
repeat 9 [ lt 90 fd 20 rt 90 fd 20 bk 20 ]
```

Aufgabe 2.15

(a) `rt 90 repeat 25 [repeat 4 [fd 15 lt 90 ]fd 15 ]`

(b) `repeat 7 [repeat 4 [fd 40 rt 90 ]fd 40 ]`

Aufgabe 2.17

(a) Zur Lösung dieser Aufgabe müssen wir drei `repeat`-Schleifen ineinander verschachteln.

Die innerste Schleife ist dir schon vielfach begegnet. Sie zeichnet ein Quadrat mit der entsprechenden Seitenlänge 20:

```
repeat 4 [ fd 20 rt 90 ]
```

Ebenfalls hast du schon gelernt, wie mehrere solcher Quadrate nebeneinander gezeichnet werden können. Mit einem weiteren `repeat`-Befehl außenrum kann das einfach bewerkstelligt werden:

```
rt 90 ... repeat 10 [ repeat 4 [ fd 20 rt 90 ] fd 20 ]
```

Dabei bewirkt der Befehl `fd 20` am Ende, dass die Schildkröte zum Startpunkt des nächsten Quadrates weiterwandert. Die Orientierung der Schildkröte wurde mit dem Befehl `rt 90` ganz am Anfang so gewählt, dass sie nach rechts schaut. Insofern genügt ein einziger Befehl `fd 20`, um ein neues Quadrat *daneben* zeichnen zu lassen.

Jetzt fehlen nur noch die einzelnen Zeilen. Dazu wird noch einmal ein `repeat`-Befehl außenrum hinzugefügt:

```
rt 90 repeat 4 [ repeat 10 [ repeat 4 [ fd 20 rt 90 ] fd 20 ]
                 bk 10*20 rt 90 fd 20 lt 90 ]
```

Wichtig ist für jede der vier Zeilen noch der Zeilenwechsel, damit die Schildkröte nach Abschluss der 10 nebeneinanderliegenden Quadrate wieder ganz links anfangen kann, aber um eine Zeile nach *unten* versetzt. Die Befehlsfolge

```
bk 10*20 rt 90 fd 20 lt 90
```

bewirkt genau dies. Du siehst an diesem Beispiel, dass wir dem Programm auch einfach Rechenanweisungen geben können (`bk 10*20`), und so nicht im Kopf ausrechnen müssen, um welche Strecke sich die Schildkröte zurückbewegen muss. Kannst du hier die Bewegungen der Schildkröte nachvollziehen?

(b) Nach genau den gleichen Überlegungen wie in Aufgabe (a) ergibt sich folgendes Programm:

```
rt 90 repeat 8 [ repeat 8 [ repeat 4 [ fd 30 rt 90 ] fd 30 ]
                 bk 8*30 rt 90 fd 30 lt 90 ]
```

(c) Auch hier folgen wir demselben Prinzip:

```
rt 90 repeat 7 [ repeat 3 [ repeat 4 [ fd 25 rt 90 ] fd 25 ]
                 bk 3*25 rt 90 fd 25 lt 90 ]
```

Aufgabe 2.19 Analog zum Beispiel im Buch verwenden wir die Befehle `pu` und `pd`, um die Schildkröte zum nächsten Quadrat wandern zu lassen, ohne eine farbige Spur zu hinterlassen. Für die nebeneinanderliegenden Quadrate mit Abstand bietet es sich natürlich an, jeweils den `repeat`-Befehl zu verwenden.

```
repeat 3 [ repeat 4 [ fd 30 rt 90 ] pu rt 90 fd 60 lt 90 pd ]
pu bk 60 lt 90 fd 5*30 rt 90 pd
repeat 2 [ repeat 4 [ fd 30 rt 90 ] pu rt 90 fd 60 lt 90 pd ]
pu bk 60 lt 90 fd 5*30 rt 90 pd
repeat 3 [ repeat 4 [ fd 30 rt 90 ] pu rt 90 fd 60 lt 90 pd ]
```

Du findest jeweils zwischen den Zeichenbefehlen der nebeneinanderliegenden Quadrate auch die Befehlsfolge zum Wechsel der Schildkröte in eine neue Zeile an die entspechende Position.

Aufgabe 2.20

```
repeat 7 [ fd 20 bk 40 fd 20 rt 90 fd 20 bk 40 pu fd 90 lt 90 pd ]
```

Aufgabe 2.21 Ein einzelnes nach links zeigendes Winkelsegment kannst du wie folgt zeichnen:

```
lt 45 fd 50 rt 90 fd 50 pu bk 50 lt 90 bk 50
```

Dabei läuft die Schildkröte im Wandermodus genau die gleiche Strecke zurück, die sie gezeichnet hat. Eine Abkürzung senkrecht nach unten müsste man zuerst ausrechnen. Sowohl die Werte 40 als auch 50 wären hier falsch.

Mit `rt 135 fd 40 lt 90 pd` wird die Schildkröte an ihre neue Startposition für das neue rechtsstehende Winkelelement gesetzt.

Beide oben erwähnten Befehlsfolgen werden fünf mal wiederholt:

```
repeat 5 [ lt 45 fd 50 rt 90 fd 50 pu bk 50
           lt 90 bk 50 rt 135 fd 40 lt 90 pd ]
```

Jetzt muss die Schildkröte an die Startposition für die nach rechts zeigenden Winkelsegmente gesetzt werden:

```
pu lt 90 fd 40 rt 90 pd
```

Weitere fünf Wiederholungen analog zu oben zeichnen anschließend auch die verbleibenden Winkelelemente.

```
repeat 5 [ rt 45 fd 50 lt 90 fd 50 pu bk 50
           rt 90 bk 50 rt 45 fd 40 lt 90 pd ]
```

Somit sieht das fertige Programm wie folgt aus:

```
repeat 5 [ lt 45 fd 50 rt 90 fd 50 pu bk 50
           lt 90 bk 50 rt 135 fd 40 lt 90 pd ]
pu lt 90 fd 40 rt 90 pd
repeat 5 [ rt 45 fd 50 lt 90 fd 50 pu bk 50
           rt 90 bk 50 rt 45 fd 40 lt 90 pd ]
```

Kontrollaufgabe 1

```
repeat 2 [ fd 150 rt 90 fd 50 rt 90 ]
```

Kontrollaufgabe 2

```
rt 90 repeat 3 [ repeat 10 [ repeat 4 [ fd 25 rt 90 ] fd 25 ]
                 bk 10*25 rt 90 fd 25 lt 90 ]
```

Kontrollaufgabe 3

```
repeat 3 [ repeat 4 [ fd 70 lt 90 fd 70 rt 90 fd 70 rt 90 ]
           pu rt 90 fd 310 lt 90 pd ]
```

Kontrollaufgabe 5

```
repeat 7 [ repeat 4 [ fd 20 lt 90 fd 20 rt 90 fd 20 rt 90 ]
           pu rt 90 fd 80 lt 90 pd ]
```

Kontrollaufgabe 6

```
rt 90 fd 50 bk 100 fd 50 lt 90 fd 350 rt 180
repeat 5 [ rt 45 fd 150 pu bk 150 lt 90 pd
           fd 150 pu bk 150 rt 45 fd 50 pd ]
```

Kapitel 3

Aufgabe 3.1 Wie erwartet zeichnet das Programm ein Quadrat mit der Seitenlänge 100.

Aufgabe 3.2

```
to QUAD200
repeat 4 [ fd 200 rt 90 ]
end
```

Aufgabe 3.4 Vor jedem Aufruf von `QUAD50` muss die Schildkröte entsprechend positioniert werden. Aus Sicht der Schildkröte wird jedes Quadrat beginnend von links unten und dann im Uhrzeigersinn gezeichnet. Versuche, die einzelnen Positionierungsschritte der Schildkröte nachzuvollziehen.

```
to QUAD50
repeat 4 [ fd 50 rt 90 ]
end

QUAD50 fd 50 QUAD50 fd 50 QUAD50 rt 90 fd 50
QUAD50 rt 90 fd 50 rt 90 fd 50 QUAD50
```

Aufgabe 3.8 Wir gehen genau gleich vor wie im erläuterten Beispiel.

```
to QUAD25
repeat 4 [ fd 25 rt 90 ]
end

to FELD1M8Q25
repeat 8 [ QUAD25 rt 90 fd 25 lt 90 ]
end

repeat 8 [ FELD1M8Q25 pu lt 90 fd 200 lt 90 fd 25 rt 180 pd ]
```

Aufgabe 3.9 Das Zeichnen einer Spalte von vier Quadraten (4×10-Feld) geschieht mit Hilfe des Programms `FELD4M1Q20`.

```
to FELD4M1Q20
repeat 4 [ QUAD20 fd 20 ]
end
```

Das Zeichnen dieser Spalten wird insgesamt zehnmal wiederholt. Dazwischen wird jeweils die Schildkröte an die richtige neue Startposition gesetzt.

```
repeat 10 [ FELD5M1Q20 pu rt 90 fd 20 rt 90 fd 80 rt 180 pd ]
```

Aufgabe 3.10

```
repeat 6 [ QUAD50 pu rt 90 fd 90 lt 90 pd ]
```

Aufgabe 3.11 Ein Programm zum Zeichnen eines Kreuzes kann wie folgt aussehen:

```
to KREUZ
fd 20 bk 40 fd 20 rt 90 fd 20 bk 40
end
```

Dieses Programm `KREUZ` wird siebenmal aufgerufen und dabei jeweils die Schildkröte an den neuen Startort positioniert:

```
repeat 7 [ KREUZ pu fd 90 lt 90 pd ]
```

Anmerkung: Prinzipiell könnte man auch die Befehlsfolge `pu fd 90 lt 90 pd` in das Programm `KREUZ` mit einbauen. In diesen Beispielen jedoch ist es ein besserer Stil, das Programm `KREUZ` wirklich nur diejenigen Befehle ausführen zu lassen, für die sein Name spricht. Wer weiss, vielleicht möchte man ja später einmal Kreuze mit anderen Abständen oder mit anderen Anordnungen zeichnen.

Aufgabe 3.12

```
to QUAD30
repeat 4 [ fd 30 rt 90 ]
end

repeat 3 [ QUAD30 pu rt 90 fd 60 lt 90 pd ]
pu bk 60 lt 90 fd 5*30 rt 90 pd
repeat 2 [ QUAD30 pu rt 90 fd 60 lt 90 pd ]
pu bk 60 lt 90 fd 5*30 rt 90 pd
repeat 3 [ QUAD30 pu rt 90 fd 60 lt 90 pd ]
```

und

```
to CHKREUZ70
repeat 4 [ fd 70 lt 90 fd 70 rt 90 fd 70 rt 90 ]
end

repeat 3 [ CHKREUZ70 pu rt 90 fd 310 lt 90 pd ]
```

Aufgabe 3.13

```
to BAUM
rt 90 fd 50 bk 100 fd 50 lt 90 fd 350 rt 180
repeat 5 [ rt 45 fd 150 pu bk 150 lt 90 pd
           fd 150 pu bk 150 rt 45 fd 50 pd ]
end

repeat 3 [ BAUM pu fd 100 lt 90 fd 300 pd lt 90 ]
```

Aufgabe 3.14 Das folgende Programm zeichnet zwei Linien im Abstand von zwei Schritten. Dieses Mal entsteht nicht eine fette Linie, sondern zwei getrennte, einzelne dünne Linien.

```
fd 100 rt 90
pu fd 2 rt 90 pd
fd 100 rt 180
```

Aufgabe 3.15

```
to FETT200M4
fd 200 rt 90
fd 1 rt 90
fd 200 lt 90
fd 1 lt 90
fd 200 rt 90
fd 1 rt 90
fd 200 lt 180
end
```

Aufgabe 3.16 Da die fette Linie ja bereits um 1 breiter ist als eine normale Linie, würde sie beim Zeichnen des Quadrates mittels `repeat 100` auch um 1 rechts überstehen. Aus diesem Grund dürfen nur 99 Wiederholungen stattfinden, damit das Quadrat genau die Breite 100 hat.

Aufgabe 3.18 Es zeichnet ebenfalls eine fette Linie, genau wie `FETT100`, nur dass die Schildkröte dabei zwei separate Linien von unten nach oben zeichnet, anstatt in einem Durchlauf wie bei `FETT100`.

Aufgabe 3.19 Zuerst braucht es noch jeweils Programme zum Zeichnen fetter Linien der Längen 50, 75 und 150:

```
to FETT50
fd 50 rt 90
fd 1 rt 90
fd 50 rt 180
end

to FETT75
fd 75 rt 90
fd 1 rt 90
fd 75 rt 180
end

to FETT150
fd 150 rt 90
fd 1 rt 90
fd 150 rt 180
end
```

Analog zum Beispiel im Buch lassen sich nun die Programme für die jeweiligen schwarzen Quadrate schreiben:

```
to SCHW50
repeat 50 [ FETT50 ]
end

to SCHW75
repeat 75 [ FETT75 ]
end

to SCHW150
repeat 150 [ FETT150 ]
end
```

Aufgabe 3.22

```
to SPALTEA
repeat 2 [ QUAD100 fd 100 SCHW100 lt 90 fd 100 rt 90 fd 100 ]
end

to SPALTEB
repeat 2 [ SCHW100 lt 90 fd 100 rt 90 fd 100 QUAD100 fd 100 ]
end

to SCHACH4SPALTEN
repeat 2 [ SPALTEA pu bk 400 rt 90 fd 100 lt 90 pd
           SPALTEB pu bk 400 rt 90 fd 100 lt 90 pd ]
end
```

Aufgabe 3.24 Die Strukturen sehen gleich aus wie in **??** bzw. **??**, nur dass es anstelle `ZEILEA` hier `SPALTEA` und anstelle `ZEILEB` hier `SPALTEB` heissen muss.

Aufgabe 3.25 Zuerst wird gemäss Aufgabenstellung ein 4×4-Feld gezeichnet. Durch Verschachtelung zweier Schleifen wird ein solches Feld zeilenweise gezeichnet:

```
repeat 4 [ repeat 4 [ QUAD100 rt 90 fd 100 lt 90 ]
           pu lt 90 fd 400 rt 90 fd 100 pd ]
```

Um anschließend jedes zweite Feld schwarz einzufärben, müssen wir unterscheiden, ob das am weitesten links stehende Feld in der aktuellen Zeile weiss oder schwarz ist. Für beide Fälle schreiben wir am einfachsten ein Unterprogramm:

```
to SCHWARZWEISS
SCHW100 pu rt 90 fd 100 lt 90 pd
end

to WEISSSCHWARZ
pu rt 90 fd 100 lt 90 pd SCHW100
end
```

Für jede Zeile wird eines der beiden Unterprogramme also jeweils zwei mal ausgeführt. Für ein 4×4-Feld werden zwei Zeilenpaare erstellt. Gesamthaft sieht das Hauptprogramm mit den verschachtelten Schleifen dann wie folgt aus:

```
to SCHACH4M4
repeat 4 [ repeat 4 [ QUAD100 rt 90 fd 100 lt 90 ]
           pu lt 90 fd 400 rt 90 fd 100 pd ]
pu bk 400 pd
repeat 2 [ repeat 2 [ SCHWARZWEISS ]
           pu lt 90 fd 400 rt 90 fd 100 pd
           repeat 2 [ WEISSSCHWARZ ]
           pu lt 90 fd 400 rt 90 fd 100 pd ]
end
```

Kontrollaufgabe 1

```
to SCHW50M150
repeat 50 [ FETT150 ]
end
```

Kontrollaufgabe 2

```
to QUAD70
repeat 4 [ fd 70 rt 90 ]
end

to FELD1M5Q70
repeat 5 [ QUAD70 rt 90 fd 70 lt 90 ]
end

repeat 4 [ FELD1M5Q70 pu lt 90 fd 5*70 lt 90 fd 70 rt 180 pd ]
```

Kontrollaufgabe 3

```
repeat 4 [ SCHW100 fd 100 ]
pu rt 180 fd 100 lt 90
pd repeat 3 [ SCHW100 fd 100 ]
```

Kontrollaufgabe 5 In dieser Beispiellösung wird gleich vorgegangen wie in **??**.

```
to SCHWARZWEISS50
SCHW50 pu rt 90 fd 50 lt 90 pd
end

to WEISSSCHWARZ50
pu rt 90 fd 50 lt 90 pd SCHW50
end

to SCHACH2M6
repeat 2 [ repeat 6 [ QUAD50 rt 90 fd 50 lt 90 ]
           pu lt 90 fd 300 rt 90 fd 50 pd ]
pu bk 100 pd
repeat 3 [ SCHWARZWEISS50 ]
pu lt 90 fd 300 rt 90 fd 50 pd
repeat 3 [ WEISSSCHWARZ50 ]
pu lt 90 fd 300 rt 90 fd 50 pd
end
```

Kontrollaufgabe 8 Zuerst braucht es Unterprogramme zum Zeichnen fetter Linien der Länge 70 und darauf basierend zum Zeichnen eines schwarzen Quadrates der Länge 70:

```
to FETT70
fd 70 rt 90
fd 1 rt 90
fd 70 rt 180
end

to SCHW70
repeat 70 [ FETT70 ]
end
```

Ein einzelnes Kreuz wird im Unterprogramm `CHKREUZSCHWARZ70` gezeichnet und setzt ein solches Kreuz aus insgesamt fünf schwarzen 70×70-Quadraten zusammen:

```
to CHKREUZSCHWARZ70
pu rt 90 fd 70 lt 90 pd SCHW70
pu fd 70 lt 90 fd 140 rt 90 pd
SCHW70 SCHW70 SCHW70
pu fd 70 lt 90 fd 210 rt 90 pd
pu rt 90 fd 70 lt 90 pd SCHW70
pu bk 140 pd
end
```

Von diesen schwarzen Schweizerkreuzen werden drei hintereinander mit dem entsprechenden Abstand gezeichnet:

```
repeat 3 [ CHKREUZSCHWARZ70 pu rt 90 fd 170 lt 90 pd ]
```

Kapitel 4

Aufgabe 4.1

(a)

```
repeat 5 [ fd 180 rt 72 ]
```

(b)

```
repeat 12 [ fd 50 rt 30 ]
```

(c)

```
repeat 4 [ fd 120 rt 90 ]
```

(d)

```
repeat 6 [ fd 100 rt 60 ]
```

(e)

```
repeat 3 [ fd 200 rt 120 ]
```

(f)

```
repeat 18 [ fd 20 rt 20 ]
```

Aufgabe 4.3 Alle Programme zeichnen Kreise verschiedener Größe.

Aufgabe 4.4 Die Summe aller Einzelschritte (`fd`) muss möglichst klein sein. Das können wir durch einen kleinen Wert hinter `fd` erreichen, oder durch weniger `repeat`-Durchgänge.

So bewirken z. B. `repeat 360 [fd 0.1 rt 1 ]` oder `repeat 36 [fd 1 rt 10 ]` beide einen sehr kleinen Kreis.

Aufgabe 4.7

```
to ECK12
repeat 12 [ fd 70 rt 360/12 ]
end

repeat 18 [ ECK12 rt 20 ]
```

Aufgabe 4.9

```
to KREIS2
repeat 360 [ fd 2 rt 1 ]
end

repeat 18 [ KREIS2 rt 20 ]
```

Es ergibt sich ein Bild, welches einer Blume ähnelt.

Aufgabe 4.10

```
setpencolor 13 MUST3
setpencolor 7 MUST3
```

Wird `MUST3` nach Aufruf des Befehls `setpencolor 7` ausgeführt, so bewirkt die weisse Stiftfarbe ein Löschen des vorher orange gezeichneten Musters.

Aufgabe 4.11 Den linken Kreis zeichnen wir in gelber Farbe linksherum (mittels `lt`), den rechten Kreis in blauer Farbe rechtsherum (mittels `rt`).

```
setpencolor 3 repeat 360 [ fd 2 rt 1 ]
setpencolor 4 repeat 360 [ fd 2 lt 1 ]
```

Kontrollaufgabe 1

(a)

```
repeat 12 [ fd 25 rt 30 ]
```

Umfang: $12 \cdot 25 = 300$.

(b)

```
repeat 7 [ fd 50 rt 360/7 ]
```

Umfang: $7 \cdot 50 = 350$.

(c)

```
repeat 3 [ fd 200 rt 120 ]
```

Umfang: $3 \cdot 200 = 600$.

Kontrollaufgabe 3 Es wird vier mal hintereinander jeweils ein schwarzer Strich (eine Seite des Quadrates) und der rote Kreis am Ende des Striches gezeichnet:

```
repeat 4 [ setpencolor 0 fd 200 rt 45
           setpencolor 1 repeat 180 [ fd 3 lt 2 ] rt 45 ]
```

Kontrollaufgabe 4 Ohne Zuhilfenahme der Mathematik ist es schwierig herauszufinden, wie weit die Schildkröte nach dem Zeichnen eines vollen Kreises nach rechts wandern muss, um den nächsten Kreis zu zeichnen. Folgender Trick hilft dabei, auf diese Rechnung zu verzichten.

Zuerst werden vier obere Halbkreise nebeneinander gezeichnet. Jedes mal, wenn die Schildkröte einen Halbkreis abgeschlossen hat, dreht sie sich um 180°, um den nächsten Halbkreis zu zeichnen. Genau gleich wird bei der unteren Hälfte der Zeichnung vorgegangen. Aus den zwei zusammengesetzten Halbkreisen ergibt sich dann das Bild wie in der Aufgabenstellung.

```
repeat 2 [ repeat 4 [ repeat 180 [ fd 1 rt 1 ] rt 180 ] rt 180 ]
```

Kontrollaufgabe 6 Das folgende Unterprogramm zeichnet zwei Fenster und eine Haustüre und kehrt mit der Schildkröte anschliessend wieder an die Ursprungsposition des aktuellen Sechsecks zurück:

```
to FENSTERTUERE
pu fd 100 rt 60 fd 100 rt 120 fd 50 rt 90 fd 25
pd repeat 4 [ fd 25 lt 90 ]
pu rt 180 fd 50
pd repeat 4 [ fd 25 rt 90 ]
pu rt 180 fd 25 lt 90 fd 100 rt 90 fd 15
pd rt 90 fd 50 rt 90 fd 30 rt 90 fd 50
pu rt 90 fd 15 rt 90 fd 150 rt 120
repeat 4 [ fd 100 rt 60 ] pd
end
```

Ausgehend von dieser Position werden die unteren beiden Linien des Sechsecks mit dem Radiergummimodus gelöscht:

```
to VERSLOESCHEN
pe rt 120 fd 100 lt 60 fd 100 lt 60 penpaint
end
```

Mit Hilfe dieser Unterprogramme und dem Unterprogramm ECK6L100 aus **??** können beliebig viele Reihenhäuser nebeneinander gezeichnet werden:

```
repeat 4 [ ECK6L100 FENSTERTUERE VERSLOESCHEN ]
```

Kontrollaufgabe 7 Zuerst erstellen wir ein Unterprogram WABE50 zum Zeichnen einer einzelnen Wabe mit Seitenlänge 50:

```
to WABE50
repeat 6 [ fd 50 rt 60 ]
end
```

Das folgende Unterprogramm WABESCHRITT50 soll den Schritt vereinfachen, die Schildkröte an die Startposition der benachbarten Wabe zu setzen. Dazu wird im Wandermodus (pu) zwei Kanten der Wabe entlanggelaufen.

```
to WABESCHRITT50
pu repeat 2 [ fd 50 rt 60 ] lt 120 pd
end
```

Im Hauptprogramm werden jetzt in vier Schritten die einzelnen Zeilen der Waben gezeichnet (Orientierung von unten links nach oben rechts). Dazwischen wird die Schildkröte für die darauffolgende Zeile jeweils richtig positioniert:

```
setpencolor 3
rt 30 repeat 3 [ WABE50 WABESCHRITT50 ]
rt 180 repeat 3 [ WABESCHRITT50 ] lt 120 WABESCHRITT50
lt 60 repeat 4 [ WABE50 WABESCHRITT50 ]
rt 180 repeat 4 [ WABESCHRITT50 ] lt 120 WABESCHRITT50
lt 60 repeat 4 [ WABE50 WABESCHRITT50 ]
rt 180 repeat 3 [ WABESCHRITT50 ] lt 120 WABESCHRITT50
lt 60 repeat 3 [ WABE50 WABESCHRITT50 ]
```

Kontrollaufgabe 8

```
repeat 180 [ fd 2 rt 2 ]
repeat 180 [ fd 3 rt 2 ]
repeat 180 [ fd 4 rt 2 ]
repeat 180 [ fd 5 rt 2 ]
```

Kapitel 5

Aufgabe 6.1 Nach dem Befehl `QUADRAT 70` befindet sich die Zahl 70 im Register. Nach der Durchführung des Programms verbleibt die Zahl 130.

Aufgabe 6.2

```
QUADRAT 50 QUADRAT 99 QUADRAT 123 QUADRAT 177
```

Aufgabe 6.3 Es ergibt eine Art „Spielbrett“ (z. B. des Spiels „Mühle“), welches um 45° gedreht ist, also auf der Spitze steht.

Die folgende Befehle würden das zweite gegebene Programm richtig abschliessen:

```
repeat 4 [ QUADRAT 70 QUADRAT 90 QUADRAT 110 rt 90 ]
```

Aufgabe 6.5

```
to FUENF :L
repeat 5 [ fd :L rt 72 ]
end
```

Aufgabe 6.6

```
VIELECK 10 VIELECK 20 VIELECK 30 VIELECK 40 VIELECK 50
```

Wir stellen fest, dass die Vielecke mit steigender Anzahl Ecken auch wesentlich grösser werden. Lies weiter, um zu erfahren, wie man auch die Grösse der Vielecke mit Hilfe von Parametern einstellen kann.

Aufgabe 6.8

```
to FETT :L
fd :L rt 90
fd 1 rt 90
fd :L rt 180
end
```

Aufgabe 6.9

```
to HAUSKLEIN :X
repeat 5 [ fd :X rt 90 ]
lt 60 fd :X rt 120 fd :X
end
```

Aufgabe 6.10

```
to HAUSGROSS :X
repeat 2 [ fd 3*:X/4 rt 90 fd :X rt 90 ]
fd 3*:X/4 rt 30 fd :X rt 120 fd :X rt 30
pu fd :X/8 rt 90 fd :X/8 pd
repeat 4 [ fd :X/8 lt 90 ]
pu fd 5*:X/8 pd
repeat 4 [ fd :X/8 lt 90 ]
pu lt 90 fd 5*:X/8 lt 90 bk :X/4 fd 5*:X/12 pd
lt 90 fd :X/3 rt 90 fd :X/6 rt 90 fd :X/3
end
```

Aufgabe 6.11

```
to TREPPEHINUNTER :ANZ
rt 90 repeat :ANZ [ fd 40 rt 90 fd 40 lt 90 ]
end

to TREPPEHINAUF :ANZ
repeat :ANZ [ fd 20 rt 90 fd 20 lt 90 ]
end
```

Aufgabe 6.12

```
to RADZAEHNE :ANZ
repeat :ANZ [ repeat 2 [ fd 20 rt 90 ] repeat 2 [ fd 20 lt 90 ] ]
end
```

Aufgabe 6.13 Das Rondell-Muster können wir einfach erstellen, wenn wir erkennen, dass es sich aus vier Viertelteilen zusammensetzt. Jedes dieser Teile besteht aus den zwei kleinen Kreisen und einem grossen Viertelkreis. Setzen wir diese vier Teile zusammen, erhalten wir das entsprechende Muster.

```
to RONDELL :UMFANG
repeat 4 [ repeat 360 [ fd :UMFANG/360/3 rt 1 ]
           repeat 360 [ fd :UMFANG/360/3 lt 1 ]
           repeat 90 [ fd :UMFANG/360 rt 1 ] ]
end
```

Aufgabe 6.14 Analog zur vorherigen Aufgabe setzen sich die Muster hier jeweils auch aus drei Teilen zusammen, die sich einfach mit einer `repeat`-Schleife wiederholen lassen.

```
to DREIECKA :X
lt 90 repeat 3 [ repeat 3 [ fd :X/4 lt 120 ] rt 120 fd :X ]
end

to DREIECKB :X
lt 90 repeat 3 [ rt 60 bk :X/8 repeat 3 [ fd :X/4 lt 120 ]
                 fd :X/8 rt 60 fd :X ]
end
```

Aufgabe 6.15

```
to KOPF2PARAM :UMGROSS :UMKLEIN
repeat 2 [ repeat 180 [ fd :UMGROSS/360 rt 1 ]
           repeat 360 [ fd :UMKLEIN/360 lt 1 ] ]
end
```

Aufgabe 6.16

```
to RONDELL2PARAM :UMGROSS :UMKLEIN
repeat 4 [ repeat 360 [ fd :UMKLEIN/360 rt 1 ]
           repeat 360 [ fd :UMKLEIN/360 lt 1 ]
           repeat 90 [ fd :UMGROSS/360 rt 1 ] ]
end
```

Aufgabe 6.17

```
to DREIECKA2PARAM :XGROSS :XKLEIN
lt 90 repeat 3 [ repeat 3 [ fd :XKLEIN lt 120 ] rt 120 fd :XGROSS ]
end

to DREIECKB2PARAM :XGROSS :XKLEIN
lt 90 repeat 3 [ rt 60 bk :XKLEIN/2 repeat 3 [ fd :XKLEIN lt 120 ]
                 fd :XKLEIN/2 rt 60 fd :XGROSS ]
end
```

Aufgabe 6.18

```
to RECHTROT :HOR :VER
setpencolor 1
repeat :HOR [ FETT :VER ]
end
```

Aufgabe 6.21

```
to RECHTFARBE :HOR :VER :FARBE
setpencolor :FARBE
repeat :HOR [ FETT :VER ]
end
```

Aufgabe 6.22

```
to TURM :BREITE :HOEHE
pu rt 90 fd 2*:BREITE/7 lt 90 pd
fd 4*:HOEHE/7 lt 90 fd 2*:BREITE/7 rt 90 fd 3*:HOEHE/7
repeat 3 [ rt 90 fd :BREITE/7 rt 90 fd :HOEHE/7
           lt 90 fd :BREITE/7 lt 90 fd :HOEHE/7 ] rt 90
fd :BREITE/7 rt 90 fd 3*:HOEHE/7 rt 90
fd 2*:BREITE/7 lt 90 fd 4*:HOEHE/7
pu lt 90 fd 4*:BREITE/7 lt 90 pd
end

to TUERME :BREITE :HOEHE :ANZAHL
repeat :ANZAHL [ TURM :BREITE :HOEHE ]
end
```

Kontrollaufgabe 1

```
to WIPPE :x
fd :X lt 90 fd :X repeat 4 [ fd :X rt 90 ]
lt 180 fd 2*:X repeat 4 [ fd :X lt 90 ]
end
```

Kontrollaufgabe 2

```
to FELDI :I
repeat 2*:I [ repeat 2*:I [ repeat 3 [ fd 50 rt 90 ] rt 90 ] lt 90
              fd 50*2*:I rt 90 fd 50 ]
end
```

Kontrollaufgabe 3

```
to BAUMXYZ :X :Y :Z :FARBE
setpencolor :FARBE
lt 90 fd 50 bk 100 fd 50 rt 90 fd 350 rt 180
rt 45 fd :X bk :X lt 90 fd :X bk :X rt 45
fd 100
rt 45 fd :Y bk :Y lt 90 fd :Y bk :Y rt 45
fd 100
rt 45 fd :Z bk :Z lt 90 fd :Z bk :Z rt 45
end
```

Kontrollaufgabe 4

```
to RECHTP1P2 :P1 :P2
repeat 2 [ fd :P1*:P2 rt 90 fd 2*:P1 rt 90 ]
end
```

Kontrollaufgabe 5 Wir verwenden das Unterprogramm RECHT zum Zeichnen eines Rechteckes mit parametrisierbarer Breite und Höhe.

```
to QQQ :ANZAHL :LANG :WINK
repeat :ANZAHL [ RECHT :LANG :LANG rt :WINK ]
end
```

Kontrollaufgabe 6

```
to QQQVOR :ANZAHL :LANG :WINK :VOR
repeat :ANZAHL [ RECHT :LANG :LANG rt :WINK fd :VOR ]
end
```

Kapitel 6

Aufgabe 7.1 Die Schildkröte fährt nach dem Zeichnen der Quadrate weiter nach vorne, bevor sie um 10 Grad abdreht und mit dem neuen Quadrat beginnt. Somit wird der Bogen grösser, auf dem sich die Quadrate befinden.

Aufgabe 7.2 Beim Testen mit unterschiedlichen Parametern stellt man fest, dass die Quadrate unterschiedliche Seitenlängen aufweisen. Der Wert des Parameters `:GR` wird also offensichtlich an das Unterprogramm weitergegeben.

Aufgabe 7.3

```
to MUS4 :GR :A :W
repeat :W [QUADRAT :GR fd :A rt 10 ]
end
```

Aufgabe 7.4

```
to MUS1RECHT
repeat 20 [RECHT 100 50  fd 30 rt 10 ]
end

to MUS2RECHT :HOR :VER
repeat 20 [RECHT  :HOR :VER  fd 30 rt 10 ]
end

to MUS3RECHT :HOR :VER :A
repeat 20  [RECHT  :HOR :VER  fd :A rt 10 ]
end
```

Das Programm `MUS2RECHT` hat 2 Parameter und das Programm `MUS3RECHT` hat 3 Parameter

Aufgabe 7.5 Beim Ausführen von `BLATT1 20 BLATT1 40 BLATT1 60 BLATT1 80 BLATT1 100` stellt man fest, dass der Radius des Bogens immer gleich bleibt, doch die Bogenlänge verändert sich.

Aufgabe 7.6 Wenn der Parameter `:LANG` fest bleib, dann haben alle Blätter dieselbe Form, sind aber unterschiedlich lang. Wenn der Parameter `GROSS` fest bleib, dann sind die kleineren Blätter schmaler als die grösseren.

Aufgabe 7.8 Ein mögliches Programm wäre:

```
to MUSTER :ANZAHL :LANG :GROSS
repeat :ANZAHL [BLUMEN 10 :LANG :GROSS fd :LANG*:GROSS rt 360/:ANZAHL]
end
```

Aufgabe 7.9

```
to BLU2 :ANZAHL :LANG :GROSS :F1 :F2
setpencolor :F1
BLUMEN :ANZAHL :LANG :GROSS
rt 5
setpencolor :F2
BLUMEN :ANZAHL :LANG :GROSS
rt 5
setpencolor :F1
MUS3 :LANG :ANZAHL
end
```

Aufgabe 7.10 Das Programm QUADRAT ist bereits bekannt:

```
to QUADRAT :GR
repeat 4 [fd :GR rt 90]
end
```

Das Programm QUADRAT wird benötigt, um das Programm ZEILE zu erstellen:

```
to ZEILE :GR :Y
repeat :Y [QUADRAT :GR rt 90 fd :GR lt 90]
end
```

Das Programm FELD entsteht nun aus dem Programm ZEILE:

```
to FELD :GR :Y :X
repeat :X [ZEILE :GR :Y lt 90 fd :Y*:GR rt 90 fd :GR]
end
```

Aufgabe 7.11 Das Programm zum Zeichnen fetter Linien kann so aussehen:

```
to FETTBEL :GR
fd :GR rt 90 fd 1 rt 90 fd :GR rt 180
end
```

Daraus kann folgendes Programm gemacht werden:

```
to SCHWBEL :GR
repeat :GR [FETTBEL :GR]
end
```

Das Programm QUADRAT ist bereits bekannt:

```
to QUADRAT :GR
repeat 4 [fd :GR rt 90]
end
```

Aus QUADRAT und SCHWBEL kann man die zwei Programme ZEILE1 und ZEILE2 aufbauen:

```
to ZEILE1 :GR :I
repeat :I [SCHWBEL :GR QUADRAT :GR rt 90 fd :GR lt 90]
fd :GR lt 90 fd :GR*:I*2 rt 90
end

to ZEILE2 :GR :I
repeat :I [QUADRAT :GR  rt 90 fd :GR lt 90 SCHWBEL :GR]
fd :GR lt 90 fd :GR*:I*2 rt 90
end
```

Aus diesen beiden ergibt sich dann das Programm SCHACHBEL:

```
to SCHACHBEL1 :GR :I
repeat :I [ZEILE1 :GR :I ZEILE2 :GR :I]
end
```

Aufgabe 7.12 Bei beiden Eingaben für BLUMEN3 wird eine Blume mit drei verschiedenen Blättergrössen gezeichnet.

Aufgabe 7.13

(a)

	0	1	2	3
ANZAHL	12	12	12	12
L1	100	100	100	100
L2	100	100	100	100
L3	100	100	100	100
G1	1	1	1	1
G2	2	2	2	2
G3	3	3	3	3
LANG	0	100	100	100
GROSS	0	1	2	3

(b)

	0	1	2	3
ANZAHL	15	15	15	15
L1	60	60	60	60
L2	80	80	80	80
L3	100	100	100	100
G1	2	2	2	2
G2	2	2	2	2
G3	2	2	2	2
LANG	0	60	80	100
GROSS	0	2	2	2

(c)

	0	1	2	3
ANZAHL	24	24	24	24
L1	90	90	90	90
L2	100	100	100	100
L3	110	110	110	110
G1	0.5	0.5	0.5	0.5
G2	1	1	1	1
G3	2.5	2.5	2.5	2.5
LANG	0	90	100	110
GROSS	0	0.5	1	2.5

Aufgabe 7.14

	0	1	2	3	4
Q1	50	50	50	50	50
Q2	75	75	75	75	75
Q3	100	100	100	100	100
Q4	125	125	125	125	125
GR	0	50	75	100	125

Aufgabe 7.16 Das Programm baut auf dem schon bekannten Programm FETTBEL auf:

```
to FETTBEL :HOCH
fd :HOCH rt 90 fd 1 rt 90 fd :HOCH rt 180
end

to QUG :HOCH :BREIT
repeat :BREIT [FETTBEL :HOCH ]
end

to QUG3 :HOCH1 :HOCH2 :HOCH3 :BREIT1 :BREIT2 :BREIT3
QUG :HOCH1 :BREIT1 rt 90 pu fd 10 pd lt 90
QUG :HOCH2 :BREIT2 rt 90 pu fd  10 pd lt 90
QUG :HOCH3 :BREIT3 rt 90 pu fd  10 pd lt 90
end
```

Aufgabe 7.19

	global	lokal
PYR	Q1,Q2,Q3,Q4	GR
QU4	G1,G2, G3, G4	GR
QUG	HOCH1, HOCH2, HOCH3, BREIT1, BREIT2, BREIT3	HOCH, BREIT
VE5	E1, E2, E3, E4, E5	ECKE

Aufgabe 7.20

	0	1	2
A	100	100	100
B	200	200	200
HOR	0	100	200
VAR	0	200	100

Aufgabe 7.21 Mit Hilfe des Programmes FETTBEL wird das Programm ROTRECHT erstellt:

```
to FETTBEL :HOCH
fd :HOCH rt 90 fd 1 rt 90 fd :HOCH rt 180
end

to ROTRECHT :HOR :VER
setpencolor 1
repeat :HOR [FETTBEL :VER]
end
```

Das Programm CHKREUZROT besteht hauptsächlich aus Wiederholungen des Programmes ROTRECHT. Die restlichen Befehl neben den Aufrufen von ROTRECHT dienen dem richtigen Positionieren der Schildkröte, bevor ein neues Rechteck gezeichnet wird.

```
to CHKREUZROT :A :B :C
repeat 4 [ROTRECHT :C :B+2*:C fd :B+2*:C lt 90 fd :C lt 180]
fd :C rt 90 fd :C lt 90
repeat 4 [ROTRECHT (:B-:A)/2 (:B-:A)/2
          pu fd :B lt 90 fd (:B-:A)/2 lt 180 pd]
end
```

Kontrollaufgabe 1

(a) Zuerst werden die beiden Unterprogramme BOGEN1 und BOGEN2 definiert. BOGEN1 bringt die Schildkröte dazu, einen Kreisbogen mit dem Winkel 60 Grad zu zeichnen. BOGEN2 dient dazu, die Schildkröte wieder an die Startposition zu befördern.

```
to BOGEN1 :UM
repeat 60 [fd :UM/ 360 rt 1]
end

to BOGEN2 :UM
pu
repeat 300 [fd :UM/360 rt 1]
pd
end
```

Das Programm BILD1 besteht im Wesentlichen aus den zwei oben genannten Unterprogrammen. Zu beachten ist die Drehung um 180 Grad und die umgekehrte Reihenfolge, in der BOGEN1 und BOGEN2 beim 2. Mal aufgerufen werden.

```
to BILD1 :UM
BOGEN1 :UM
BOGEN2 :UM
lt 180
BOGEN2 :UM
BOGEN1 :UM
end
```

(b) Das Programm zum Zeichnen dieser Figur ist praktisch gleich aufgebaut, wie BILD1, der Unterschied liegt nur im Anfangswinkel und im Winkel nach den Aufrufen von BOGEN1 und BOGEN2

```
to BILD2 :um
rt 45
BOGEN1 :UM
BOGEN2 :UM
rt 90
BOGEN2 :UM
BOGEN1 :UM
end
```

Kontrollaufgabe 3 Die ersten beiden Programme sind aus Aufgabe 6.11 bereits bekannt. Aus diesen Programmen kann nun TREPPESCHW erstellt werden.

```
to FETTBEL :LANG
fd :LANG rt 90 fd 1 rt 90 fd :LANG rt 180
end

to SCHWBEL :LANG
repeat :LANG [FETTBEL :LANG]
end

to TREPPESCHW :LANG :ANZ
repeat :ANZ [SCHWBEL :LANG fd :LANG ]
lt 90 fd :LANG rt 180
repeat :ANZ [ SCHWBEL :LANG fd :lang ]
end
```

Das Programm zeichnet zuerst ANZ Quadrate von unten links nach oben rechts. Danach wird die Schildkröte auf die richtige Position gesetzt, damit sie die gleiche Figur um 90 Grad nach rechts gedreht wiederholen kann. Das letzte Quadrat der ersten Figur und das erste Quadrat der zweiten Figur kommen dabei aufeinander zu liegen.

Kontrollaufgabe 4 Das Programm zum Zeichnen der Kreuze besteht aus den schon viel benutzen Unterprogrammen FETTBEL und SCHWBEL

```
to FETTBEL :LANG
fd :LANG rt 90 fd 1 rt 90 fd :LANG rt 180
end

to SCHWBEL :LANG
repeat :LANG [FETTBEL :LANG]
end

to KREUZEROT :GR :ANZ
setpencolor 1
repeat :ANZ
[repeat 3 [SCHWBEL :GR]
pu fd :GR*2 lt 90 fd :GR*2 rt 180 pd
repeat 3 [SCHWBEL :GR]
pu fd :GR*3 lt 90 fd :GR pd]
end
```

Das Programm zeichnet ein Rechteck der Länge 3GR und Höhe GR, danach wird das gleiche Rechteck, um 90 Grad gedreht, über das andere gezeichnet. Dieser Vorgang wird ANZ-mal wiederholt.

Kontrollaufgabe 5 Eine Methode zur Erstellung des Programmes ist die Nutzung des vorgegebenen Unterprogrammes KREIS2 und eines weiteren Programmes, hier HALBKREIS genannt, mit welchem die Schildkröte den halben Kreis nachfährt und sich danach um 180 Grad dreht, damit Sie den neuen Keis beginnen kann.

Eine andere Möglichkeit wäre die Berechnung des Radius in Abhängigkeit vom Umfang, damit ein Unterprogramm geschrieben werden kann, mit welchem die Schildkröte nach Beenden eines Kreises auf direktem Weg zum Startpunkt des neuen Kreises gelangt.

```
to KREIS2 :UM :FAR
setpencolor :FAR
repeat 360 [ fd :UM rt 1]
end

to HALBKREIS :GR
repeat 180 [ fd :GR rt 1]
rt 180
end

to KETTE :UM1 :UM2 :UM3 :UM4 :FAR1 :FAR2 :FAR3 :FAR4
KREIS2 :UM1 :FAR1
HALBKREIS :UM1
KREIS2 :UM2 :FAR2
HALBKREIS :UM2
KREIS2 :UM3 :FAR3
HALBKREIS :UM3
KREIS2 :UM4 :FAR4
```

```
HALBKREIS :UM4
end
```

Die Registerinhalte des Programmes KETTE entwickeln sich folgendermassen, wenn der Aufruf KETTE 1 2 1 3 4 7 10 12 lautet:

	0	1	2	3	4	5	6	7	8
UM1	1	1	1	1	1	1	1	1	1
UM2	2	2	2	2	2	2	2	2	2
UM3	1	1	1	1	1	1	1	1	1
UM4	3	3	3	3	3	3	3	3	3
FAR1	4	4	4	4	4	4	4	4	4
FAR2	7	7	7	7	7	7	7	7	7
FAR3	10	10	10	10	10	10	10	0	10
FAR4	12	12	12	12	12	12	12	12	12
UM	0	1	1	2	2	1	1	3	3
FAR	0	4	4	7	7	10	10	12	12
GR	0	0	1	1	2	2	1	1	3

Kontrollaufgabe 6

```
to ECK6 :GR
repeat 6 [fd :GR rt 60]
end

to WABESCHRITT :GR
pu repeat 2 [ fd :GR rt 60 ] lt 120 pd
end

to WABEN1 :GR :FAR
setpencolor :FAR
rt 30 repeat 3 [ ECK6 :GR WABESCHRITT :GR]
rt 180 repeat 3 [ WABESCHRITT :GR ] lt 120 WABESCHRITT :GR
lt 60 repeat 4 [ ECK6 :GR  WABESCHRITT :GR]
rt 180 repeat 4 [ WABESCHRITT :GR ] lt 120 WABESCHRITT :GR
lt 60 repeat 4 [ ECK6 :GR WABESCHRITT :GR ]
rt 180 repeat 3 [ WABESCHRITT :GR ] lt 120 WABESCHRITT :GR
lt 60 repeat 3 [ ECK6 :GR WABESCHRITT :GR]
end
```

Kontrollaufgabe 7

```
to QUADRAT :GR
repeat 4 [fd :gr rt 90]
end

to QUAD5 :G1 :G2 :G3 :G4 :G5 :F1 :F2 :F3 :F4 :F5
setpencolor :F1
QUADRAT :G1
setpencolor :F2
QUADRAT :G2
setpencolor :F3
QUADRAT :G3
```

```
setpencolor :F4
QUADRAT :G4
setpencolor :F5
QUADRAT :G5
end
```

Das Programm QUAD5 besteht aus fünf Aufrufen des Unterprogrammes QUADRAT mit unterschiedlichen Parametern. Zwischen jedem Aufruf von QUADRAT wird die Farbe gewechselt.

Der Aufruf von QUAD5 50 70 100 140 190 1 2 3 4 5 ergibt folgende Registerentwicklung:

	0	1	2	3	4	5	6	7	8	9	10
G1	50	50	50	50	50	50	50	50	50	50	50
G2	70	70	70	70	70	70	70	70	70	70	70
G3	100	100	100	100	100	100	100	100	100	100	100
G4	140	140	140	140	140	140	140	140	140	140	140
G5	190	190	190	190	190	190	190	190	190	190	190
F1	1	1	1	1	1	1	1	1	1	1	1
F2	2	2	2	2	2	2	2	2	2	2	2
F3	3	3	3	3	3	3	3	3	3	3	3
F4	4	4	4	4	4	4	4	4	4	4	4
F5	5	5	5	5	5	5	5	5	5	5	5
GR	0	50	50	70	70	100	100	140	140	190	190

Kontrollaufgabe 8 Das Programm für diese Aufgabe ähnelt sehr stark dem Programm QUAD5 aus Kontrollaufgabe **??** Der einzige Unterschied liegt im Unterprogramm, das hier ein Achteck statt eines Quadrates zeichnet.

```
to ECK8 :GR
repeat 8 [fd :GR rt 45]
end

to ECK8FUENF :G1 :G2 :G3 :G4 :G5 :F1 :F2 :F3 :F4 :F5
setpencolor :F1
ECK8 :G1
setpencolor :F2
ECK8 :G2
setpencolor :F3
ECK8 :G3
setpencolor :F4
ECK8 :G4
setpencolor :F5
ECK8 :G5
end
```

Kapitel 7

Aufgabe 8.1 Die Anzahl der Zeilen gibt keine Auskunft über die Anzahl Befehle die in einem Programm verwendet werden. Denn jede Zeile kann von keinem bis zu hunderten Befehlen enthalten.

Aufgabe 8.2 Die Programmlänge der **??** beträgt 12, die des Beispiels **??** beträgt 7 und die des Beispiels **??** beträgt 5.

Aufgabe 8.4 Die Länge des Musterprogrammes für **??** beträgt55, die des Programmes für **??** beträgt 51. Die Länge der selbst erstellten Programme kann abweichen.

Aufgabe 8.5

```
to FELDKURZ :GR :M :N
repeat :M [repeat :N [repeat 7 [fd :GR rt 90] rt 90]
           lt 90 fd :GR*:N rt 90 fd :GR]
end
```

Aufgabe 8.7

```
to QUAD6KURZ
repeat 6 [repeat 7 [ fd :50 rt 90] pu rt 180 fd 40 pd lt 90]
end
```

Aufgabe 8.8

```
to KREUZE3
repeat 3 [repeat 4 [fd 70 rt 90 fd 70 lt 90 fd 70 lt 90]
          pu rt 90 fd 310 lt 90 pd]
end
```

Aufgabe 8.9

```
to SCHACH4KURZ
repeat 2 [ZEILEA fd 100 lt 90 fd 400 rt 90
          ZEILEB fd 100 lt 90 fd 400 rt 90]
end
```

Aufgabe 8.11

```
to PYRKURZ :ANZ
repeat 2 [repeat 2 [repeat :ANZ [fd 5 rt 90 fd 5 lt 90] rt 90]
          lt 90 fd 10 lt 90]
pe lt 90 fd 10
end
```

Aufgabe 8.12

(a) Die Berechnungskomplexität beträgt 43.

(b) Die Berechnungskomplexität beträgt 110.

Aufgabe 8.13 Eine mögliche Variante für ein Programm mit weniger als 60 Grundbefehlen sieht so aus:

```
to LEITERKURZ
repeat 5 [fd 20 rt 90 fd 20 rt 90 fd 20 lt 90 fd 20 lt 90]
fd 20 lt 90 fd 200 lt 90 fd 20 lt 90 fd 200
end
```

Aufgabe 8.14

```
to SECHSECKE4
repeat 4 [repeat 3 [fd 100 rt 60] fd 100 rt 180]
lt 120
repeat 4 [ fd 100 rt 60 fd 100 lt 60]
end

to SECHSECKE10
repeat 10 [repeat 3 [fd 100 rt 60] fd 100 rt 180]
lt 120
repeat 10 [fd 100 rt 60 fd 100 lt 60]
end
```

Diese zwei Programme lösen die Aufgabe mit 49, beziehungsweise 121 Grundbefehlen.

Aufgabe 8.16 Dieses Programm kommt mit 19 Befehlen aus:

```
to TREPPEEFF
fd 100 rt 90 fd 200 lt 90 fd 200 rt 90 fd 200 lt 90 fd 100
lt 90
fd 100 lt 90 fd 200 rt 90 fd 200 lt 90 fd 200 rt 90 fd 100
end
```

Aufgabe 8.19 Ein mögliches Programm zum Zeichnen regelmässiger Vielecke könnte so aussehen:

```
to VIELECK :ANZ :GR
repeat :ANZ [fd :GR rt 360/:ANZ]
end
```

Die Berechnungskomplexität hängt vom Parameter ANZ ab und beträgt $2 \cdot$ ANZ.

Aufgabe 8.20 Die Berechnungskomplexität dieses Programmes ist $9 \cdot$ ANZAHL.

Kontrollaufgabe 1

```
to QUADKREUZ
repeat  4 [repeat 3 [repeat 2 [fd 30 rt 90] fd 30 rt 180]rt 90 bk 120]
end
```

Dieses Programm löst die Aufgabe mit einer Länge von 9.

Kontrollaufgabe 2 Siehe Lösung zur vorherigen Kontrollaufgabe. Dieses Programm hat eine relativ kleine Berechnungskomplexität von 80.

Kontrollaufgabe 3 Wie in den beiden vorangegangenen Kontrollaufgaben kann dasselbe Programm sowohl eine kleine Programmlänge als auch eine geringe Berechnungskomplexität aufweisen.

```
to QUADKREUZ1 :ANZ :GR
repeat  4 [repeat :ANZ [repeat 2 [fd :GR rt 90] fd :GR rt 180]
           rt 90 bk :GR*(:ANZ+1)]
end
```

Kontrollaufgabe 4 Diese Programme lösen die Aufgabe mit einer Länge von 7 und mit einer Berechnungskomplexität von 188.

```
to RAHMENKURZ
repeat 4 [repeat 10 [repeat 7 [fd 25 rt 90] rt 90] lt 90]
end

to RAHMENEFF
repeat 4 [repeat  5 [fd 25 rt 90 fd 25 rt 90 fd 25 lt 90 fd 25 lt 90]
          fd 25 lt 90 fd 250 lt 90 fd 25 rt 90 bk 250]
end
```

Kontrollaufgabe 5

```
to QUAD7
repeat 2 [repeat 4 [fd 30 rt 90] pu fd 90 rt 180 pd
          repeat 4 [fd 30 rt 90] pu rt 180 fd 30 pd
          repeat 4 [fd 30 rt 90] pu bk 120 lt 90 fd 60 rt 90 pd]
repeat 4 [fd 30 rt 90] pu fd 120 pd
repeat 4 [fd 30 rt 90]
end
```

Diese Lösung hat eine Programmlänge von 33 und eine Berechnungskomplexität von 95.

Kontrollaufgabe 6 Zwei Lösungen, je eine auf Länge und auf Berechnungskomplexität minimiert, können so aussehen:

```
to RECHTSCHWEFF :HOR :VER
repeat :HOR/2 [fd :VER rt 90 fd 1 rt 90 fd :VER lt 90 fd 1 lt 90]
end

to RECHTSCHWKURZ :HOR :VER
repeat :HOR [fd :VER bk :VER rt 90 fd 1 lt 90]
end
```

Beim Programm `RECHTSCHWEFF` gilt zu beachten, dass dieses nur mit geraden Parametern für `HOR` funktioniert, da `HOR`/2 sonst keine ganze Zahl ergibt, womit der Befehl `repeat` nicht umgehen kann.

Kontrollaufgabe 7

```
to PYRSCHWKURZ :GR
repeat 2[repeat 4 [repeat :GR [fd :GR bk :GR rt 90 fd 1 lt 90] fd :GR]
         rt 90 bk :GR]
end
```

```
to PYRSCHWEFF :GR
repeat 4 [repeat :GR/2 [fd :GR rt 90 fd 1 rt 90 fd :GR lt 90 fd 1 lt 90]
          fd :GR]
bk :GR rt 90
repeat 3 [repeat :GR/2 [fd :GR rt 90 fd 1 rt 90 fd :GR lt 90 fd 1 lt 90]
          fd :GR]
end
```

Auch bei PYRSCHWEFF gilt zu beachten, dass dieses Programm nur mit geraden Parameterwerten funktioniert.

Kontrollaufgabe 8 Die Berechnungskomplexität eienes Programmes zum Zeichnen eines Vieleckes beträgt mindestens das doppelte der Anzahl Ecken. Für ein Viereck wäre also mindestens eine Berechnungskomplexität von 8 nötig.

Kontrollaufgabe 9

```
to FELD5X10EFF :GR
fd :GR*10 rt 90
repeat 5 [fd :GR rt 90 fd :GR*10 bk :GR*10 lt 90]
rt 90 fd :GR*10 rt 90
repeat 5 [fd :GR*5 rt 90 fd :GR rt 90 fd :GR*5 lt 90 fd :GR lt 90]
end

to FELD5X10KURZ :GR
repeat 10 [repeat 5 [repeat 7 [fd :GR rt 90] rt 90]
           lt 90 fd :GR*5 rt 90 fd :GR]
end
```

Kontrollaufgabe 10 Ein kurzes Programm kann so realisiert werden:

```
to ZEILEKURZ :GR
repeat 4 [repeat 7 [fd :GR rt 90] rt 90
          repeat :GR [fd :GR bk :GR rt 90 fd 1 lt 90]]
end

to SCHACH8X8KURZ :GR
repeat 4 [ZEILEKURZ :GR fd :GR*2 rt 180 ZEILEKURZ :GR rt 180]
end
```

Ein Programm mit kleiner Berechnungskomplexität kann aus den folgenden Unterprogrammen zusammengestellt werden.

```
to ZEILE1EFF :GR
repeat 4 [rt 90 fd :GR lt 90
          repeat :GR/2 [fd :GR rt 90 fd 1 rt 90 fd :GR lt 90 fd 1 lt 90]]
end

to ZEILE2EFF :GR
repeat 4 [repeat :GR/2 [fd :GR rt 90 fd 1 rt 90 fd :GR lt 90 fd 1 lt 90]
          rt 90 fd :GR lt 90]
end
```

Das Hauptprogramm sieht dann so aus:

```
to SCHACH8X8EFF :GR
lt 180 bk :GR
repeat 4 [ZEILE1EFF :GR
          fd :GR lt 90 fd :GR*8 rt 90
          ZEILE2EFF :GR
          fd :GR lt 90 fd :GR*8 rt 90]
          bk :GR*8
end
```

Zu beachten ist, dass das Programm mit geradem Wert für `GR` aufgerufen wird (siehe Kontrollaufgabe **??**).

Kontrollaufgabe 11

```
to QUAD5KURZ :GR1 :GR2
repeat 4 [fd :GR1 rt 90 fd :GR2*2+:GR1 rt 90 fd :GR1 rt 90]
end
```

Kontrollaufgabe 12

```
to QUAD5EFF :GR1 :GR2
repeat 3 [fd :GR1 rt 90 fd :GR2*2+:GR1 rt 90 fd :GR1 rt 90]
fd :GR1 rt 90 fd :GR2*2+:GR1 rt 90 fd :GR1
end
```

Es kann kein kürzeres Programm zum Zeichnen dieses Bildes geben, da die Schildkröte nie eine Linie zweimal gelaufen ist. Ebenso hat sich die Schildkröte nie gedreht, ausser es war notwendig. Es sind also nur Befehle ausgeführt worden, die unbedingt nötig waren.

Kapitel 8

Aufgabe 9.1 Beim ersten Aufruf werden 20 Quadrate mit den Seitenlängen 20, 30, 40, etc. gezeichnet, beim zweiten Aufruf gibt es 5 Quadrate mit den Grössen 100, 110, 120, 130 und 140. Beim dritten Aufruf werden 25 Quadrate mit den Grössen 10, 20, . . . , 250 gezeichnet.

Aufgabe 9.4 Die Quadrate werden kleiner wenn der Wert des Parameters `MULT` kleiner als 1 ist. Ist `MULT` grösser als 1 werden die Quadrate grösser.

Aufgabe 9.5

```
to ABB8P3NEU :GR :AN :ADD
repeat :AN [repeat 6 [fd :GR rt 90] rt 180
            make "GR :GR-:ADD]
end
```

Aufgabe 9.6 Die Lösung mit einem additiven Faktor sieht folgendermassen aus:

```
to STAPELADD :GR :ANZ :ADD
lt 90
repeat :ANZ [repeat 6 [fd :GR/2 rt 90 fd :GR/2] rt 180
             make "GR :GR-:ADD]
end
```

Mit einem multiplikativen Faktor sieht die Lösung so aus:

```
to STAPELMULT :GR :ANZ :MULT
lt 90
repeat :ANZ [repeat 6 [fd :GR/2 rt 90 fd :GR/2] rt 180
             make "GR :GR*:MULT]
end
```

Aufgabe 9.8

```
to VIELECKE :GR :ECK :ANZ :ADD
repeat :ANZ [repeat :ECK [fd :GR rt 360/:ECK]
             make "GR :GR+:ADD]
end
```

`ECK`, `ANZ` und `ADD` sind Parameter. `GR` ist kein Parameter, da sich sein Wert während des Programmes ändert.

Aufgabe 9.11

	0	1	2	3	4
A	20	20	20	20	20
B	25	25	25	25	25
C	10	10	10	10	10
X	-	625	625	-175	-175
Y	-	-	800	800	800

Wie bei der Ausführung von `T3 20 25 10` ersichtlich ist, kann das Programm auch mit negativen Zahlen umgehen. Der Aufruf `fd (-175)` ist dabei gleichbedeutend mit `bk 175`.

Aufgabe 9.12 Der einzige Parameter von `TT` ist `B`.

(a)

	0	1	2	3	4	5	6	7
A	0	5	5	5	-68	-68	-68	-68
B	10	10	10	10	10	10	10	10
X	-	-	12	12	12	12	62	62
Y	-	-	-	40	40	40	40	10/62
Z	-	-	-	-	-	10	10	10

(b)

	0	1	2	3	4	5	6	7
A	5	10	10	10	-53	-53	-53	-53
B	30	30	30	30	30	30	30	30
X	-	-	27	27	27	27	97	97
Y	-	-	-	40	40	40	40	30/97
Z	-	-	-	-	-	30	30	30

(c)

	0	1	2	3	4	5	6	7
A	-5	0	0	0	-53	-53	-53	-53
B	20	20	20	20	20	20	20	20
X	-	-	27	27	27	27	87	87
Y	-	-	-	40	40	40	40	20/87
Z	-	-	-	-	-	20	20	20

Aufgabe 9.13

```
to VIELECKE2 :ANZ
make "ECK 3
repeat :ANZ [repeat :ECK [fd 20 rt 360/:ECK]
             make "ECK :ECK+1]
end
```

Aufgabe 9.14

(a)

```
to RE1ZU1 : UM
make "HOR :UM/4
RECHT :HOR :HOR
end
```

(b)

```
to RE3ZU1 :UM
make "HOR :UM*3/8
make "VER :UM/8
RECHT :HOR :VER
end
```

Aufgabe 9.16

```
to QUADSQRT :A :B
make "C sqrt (:A*:A+:B*:B)
repeat 7 [fd :A rt 90] lt 180 pu fd 3 pd lt 90
repeat 7 [fd :B rt 90] lt 180 pu fd 3 pd lt 90
repeat 7 [fd :C rt 90] lt 180 pu fd 3 pd lt 90
end
```

Wichtig an diesem Programm ist die Zeile, in der die Variable C erstellt wird. Dabei wird der neu erlernte Befehl sqrt angewendet.

Aufgabe 9.17

```
to LINGLEICHUNG :A :B :C
make "X (:C-:B)/:A
fd :X*10
end
```

Aufgabe 9.18 Die Eingaben für den Aufruf T3 1 (-10)5 sind 1, (−10) und 5. Die Ausgaben sind 80 für X und 20 für Y sowie das gezeichnete Quadrat mit Seitenlänge 80.

Aufgabe 9.20

```
to KREISE :UM :ADD
make "FAR 1
repeat 12 [setpencolor :FAR repeat 360 [fd :UM/360 rt 1]
           make "UM :UM+:ADD make "FAR :FAR+1]
end
```

Kontrollaufgabe 1

```
to TREPPEGROSS :GR :ANZ
repeat :ANZ [fd :GR rt 90 fd :GR lt 90 make "GR :GR+5]
end
```

Kontrollaufgabe 2

```
to PYRAMIDE :GR :ANZ :ADD
repeat :ANZ [fd :GR rt 90 fd :GR lt 90 make "GR :GR-:ADD]
rt 90 make "GR :GR+:ADD
repeat :ANZ [fd :GR rt 90 fd :GR lt 90 make "GR :GR+:ADD]
end
```

Kontrollaufgabe 3 Das Programm SPIR erstellt eine Spirale, an der Kreise fixiert sind. Alle drei Variablen sind Parameter.

	0	1	2	3	4	5	6	7	8	9	10
UM	50	70	100	140	190	250	320	400	490	590	700
ADD	20	30	40	50	60	70	80	90	100	110	120
AN	10	10	10	10	10	10	10	10	10	10	10

Kontrollaufgabe 4

```
to LINGL :A :B :C :D
make "X (:D-:B)/(:A-:C)
repeat 4 [fd :X*5 rt 90]
end
```

Parameter sind A, B, C und D. Das Programm wird abgebrochen, wenn der Aufruf LINGL 4 50 4 100 erfolgt, da dadurch :A-:C genau Null ergibt und durch Null nicht geteilt werden darf.

Kontrollaufgabe 7

```
to KREISINKREIS :UM :ANZ
repeat :ANZ [repeat 360 [fd :UM/360 rt 1]
             make "UM :UM*1.2]
end
```

Kontrollaufgabe 8

```
to HALBKREISE :ANZ
make "D1 100
make "D2 120
make "D3 :D1+:D2
make "UM1 :D1* 3.1415926535
make "UM2 :D2* 3.1415926535
make "UM3 :D3* 3.1415926535
repeat 180 [fd :UM1/360 rt 1] pu
repeat 180 [fd :UM1/360 rt 1] pd
repeat 180 [fd :UM2/360 rt 1] pu
repeat 180 [fd :UM2/360 rt 1] pd
repeat :ANZ [repeat 180 [fd :UM3/360 rt 1] pu
             repeat 180 [fd :UM3/360 rt 1] pd
             make "D1 :D2 make "D2 :D3
             make "D3 :D1+:D2 make "UM3 :D3* 3.1415926535 ]
end
```

Kontrollaufgabe 9

```
to ABB8P12 :GR
repeat 3 [fd :GR make "GR :GR/2 fd :GR
          repeat 3 [make "GR :GR /2
                    repeat 4 [fd :GR rt 90]
                    lt 90 make "GR :GR*2 fd :GR]
          make "GR :GR*2]
fd :GR
end
```

Kapitel 9

Aufgabe 10.1

	0	1	2	3	4	5
A	100	100	100	100	100	100
B	200	200	200	200	200	200
VER	0	100	200	200	100	200
HOR	0	200	100	100	200	100

Aufgabe 10.2 Die globalen Parameter von AUGE sind AN und NACH. UM ist eine globale Variable. Der einzige lokale Parameter ist LA.

Aufgabe 10.3 Der einzige globalen Parameter von SPIR ist AN. UM und ADD sind lokale Variablen. Der einzige lokale Parameter ist LA.

Aufgabe 10.4 Weitere Beispiele wären MUS3 mit dem Unterprogramm QUADRAT sowie BLUMEN mit dem Unterprogramm BLATT.

Aufgabe 10.5

	0	1	2	3
UM(RE2ZU1)	600	600	600	600
HOR(RE2ZU1)	-	300	300	300
VER(RE2ZU1)	-	-	150	150
HOR(RECHT1)	0	0	0	300
VER(RECHT)	0	0	0	150

Aufgabe 10.6 Die globalen Variablen des Programmes VIELQ1 sind GR(VIELQ1), AN und ST. Die einzige lokale Variable ist GR(QUADRAT)

Die folgende Tabelle zeigt die Entwicklung des Aufrufs VIELQ 20 5 10. Die Spalten zeigen den Zustand des Registers nach dem Durchlauf einer repeat-Schleife:

	0	1	2	3	4	5
GR(VIELQ1)	20	30	40	50	60	70
AN(VIELQ1)	5	5	5	5	5	5
ST(VIELQ1)	10	10	10	10	10	10
GR(QUADRAT)	0	20	30	40	50	60

Aufgabe 10.7

Befehl	0	1	2	3	4	5	6	7
GR(TEST1)	100	100	100	100	100	100	100	100
GR(TEST2)	0	0	0	100	100	200	200	200

Aufgabe 10.8

```
to TESTEIGEN1 :GR
TESTEIGEN2 :GR
repeat 4 [fd :GR rt 90]
end

to TESTEIGEN2 :GR
repeat 7 [fd :GR rt 90] lt 180 pu fd 3 lt 90 pd
make "GR :GR*2
end

to TESTEIGEN3 :GR
repeat 7 [fd :GR rt 90] lt 180 pu fd 3 lt 90 pd
make "GR :GR*2
repeat 4 [fd :GR rt 90]
end
```

Das Programm TESTEIGEN1 verwendet TESTEIGEN2 als Unterprogramm mit einem Parameter mit demselben Namen. Trotz dem selben Namen haben die Variablen am Ende des Programmes einen anderen Wert, da sonst nicht zwei gleich grosse Quadrate gezeichnet worden wären.

Das Programm TESTEIGEN3 dient zur Demonstration, was passiert wäre, wenn einfach der Inhalt von TESTEIGEN2 in TESTEIGEN1 eingefügt worden wäre.

Aufgabe 10.9 Im Programm SPIRT wird der Umfang der Kreise nicht vergrössert, da der make-Befehl um ADD zu vergrössern in einem Unterprogramm ausgeführt wird und deshalb auf die Variable des Unterprogrammes wirkt. Die Variable ADD im Hauptprogrammm bleibt gleich gross.

Beim Aufruf von TEST3 wird der Wert der Variablen ADD(SPIRT) in das Register der Variablen ADD(TEST3) kopiert. Durch den make-Befehl wird nur diese Kopie verändert.

Registerinhalte für den Aufruf SPIR 10 20 1:

Befehle	0	1	2	3	4	5	6
UM	10	10	10	10	10	30	30
ADD	20	20	20	20	20	20	30
AN	1	1	1	1	1	1	1
LA	0	0	1/36	1/36	1/36	1/36	1/36

Registerinhalte für den Aufruf SPIRT 10 20 1:

Befehle	0	1	2	3	4	5	6	7
UM	10	10	10	10	10	30	30	30
ADD(SPIRT)	20	20	20	20	20	20	20	20
AN	1	1	1	1	1	1	1	1
LA	0	0	1/36	1/36	1/36	1/36	1/36	1/36
ADD(TEST)	0	0	0	0	0	0	20	30

Kontrollaufgabe 2

```
to STRASSE :AN :A
repeat :AN [SCHWDR :A  rt 90 fd :A lt 90 fd :A/2 rt 90
            QUADRAT :A rt 180]
end
```

Die Variable A(STRASSE) behält durch den ganzen Programmablauf den Wert 70. Die Variabelle A(SCHWDR) hat direkt nach dem Aufruf von SCHWDR den Wert 70, nach der Ausführung des Unterprogrammes hat sie den Wert 0, da sie bei jedem Schlaufendurchlauf um 2 verringert wird. Diese Schlaufe wird 35-mal durchlaufen, also wird 70 abgezogen, was den Wert 0 ergibt.

Kontrollaufgabe 3

```
to SCHWDR2 :A
rt 90 make "WIN 90
repeat :A [fd :A rt 180 fd 0.5 rt :WIN
           fd 1 lt :WIN make "A :A-1
           make "WIN :WIN+180]
end
```

Kontrollaufgabe 4

```
to STRASSE2 :AN :A
repeat :AN [SCHWDR :A  rt 90 fd :A lt 90 fd :A/2 rt 90
            QUADRAT :A rt 180 make "A :A+10]
end
```

Kontrollaufgabe 5

```
to SCHWDR3 :A :H
rt 90 make "WIN 90 make "A1 :A
repeat :H [fd :A rt 180 fd (:A1/:H)/2  rt :WIN
           fd 1 lt :WIN make "A :A-:A1/:H
           make "WIN :WIN+180]
end
```

Das Programm SCHWDR3 funktiniert ähnlich wie das Programm SCHWDR. Das Dreieck besteht eigentlich aus aufeinandergestapelten Linien. Jede dieser Linien ist dabei etwas kürzer als die vorherige. Die Linien werden so gekürzt, dass die letzte Linie nicht mehr von einem Punkt unterschieden werden kann.

Um ein Dreieck mit der Höhe H zu zeichnen, muss die Strecke, die als Basis verwendet wird H-mal gekürzt werden, und zwar so, dass sie am Schluss genau die Länge 0 aufweist. Aus diesem Grund wird sie in jeder der H repeat-Schleifen um den H-ten Bruchteil gekürzt.

Der Befehl fd (:A1/:H) sorgt dafür, dass die Linien zentriert gestapelt werden, da sonst ein rechtwinkliges Dreieck entstehen würde.

Die Variable A1 hat nur die Aufgabe den ursprünglichen Wert von A zu speichern, damit dieser zum korrekten Kürzen der Strecke noch zur Verfügung steht.

Kontrollaufgabe 6 HOCH, SUBL und SUBH sind Parameter von PFLANZE. Die globalen Variablen von PFLANZE sind LANG(PFLANZE), HOCH, STEP, SUBL und SUBH. Lokale Variable ist nur LANG(NADEL).

Die folgende Tabelle zeigt die Entwicklung der Register nach den ersten drei repeat-Schleifendurchläufen.

Schleifendurchlauf	0	1	2	3
LANG(PFLANZE)	100	98	96	94
HOCH	50	50	50	50
STEP	15	14.7	14.4	14.1
SUBL	2	2	2	2
SUBH	0.3	0.3	0.3	0.3
LANG(NADEL)	0	100	98	96

Kontrollaufgabe 7 Ja, es macht einen Unterschied. Die lokale Variable LANG(NADEL) wird zwar geändert, aber bei jedem Aufruf von NADEL wird der Wert der lokalen Variable LANG(NADEL) mit dem Wert der globalen Variablen LANG(PFLANZE) überschrieben. Aus diesem Grund bliebt beim veränderten Programm die Nadellänge immer gleich.

Kontrollaufgabe 8 Zuerst wird das Programm PFLANZE so geändert, dass die Schildkröte nach dem Zeichnen wieder an den Startpunkt zurückkehrt:

```
to PFLANZEBACK :LANG :HOCH :STEP :SUBL :SUBH
repeat :HOCH [fd :STEP NADEL :LANGZ rt 3
              make "LANG :LANG-:SUBL
              make "STEP :STEP-:SUBH]
repeat :HOCH [make "STEP :STEP+:SUBH
                   lt 3 bk :STEP]
end
```

Nun kann das Programm STRAUSS geschreiben werden. Es besteht nur darin, das Programm PFLANZEBACK aufzurufen und zwischen den Aufrufen die Farbe und den Winkel für die nächste Pflanze einzustellen.

```
to STRAUSS :LANG1 :HOCH1 :STEP1 :SUBL1 :SUBH1 :FAR1 :WIN1 :LANG2 :HOCH2
:STEP2 :SUBL2 :SUBH2 :FAR2 :WIN2 :LANG3 :HOCH3 :STEP3 :SUBL3 :SUBH3 :FAR3
:WIN3 :LANG4 :HOCH4 :STEP4 :SUBL4 :SUBH4 :FAR4 :WIN4 :LANG5 :HOCH5 :STEP5
:SUBL5 :SUBH5 :FAR5 :WIN5
setpencolor :FAR1 rt :WIN1
PFLANZEBACK :LANG1 :HOCH1 :STEP1 :SUBL1 :SUBH1
setpencolor :FAR2 rt :WIN2
PFLANZEBACK :LANG2 :HOCH2 :STEP2 :SUBL2 :SUBH2
setpencolor :FAR3 rt :WIN3
PFLANZEBACK :LANG3 :HOCH3 :STEP3 :SUBL3 :SUBH3
setpencolor :FAR4 rt :WIN4
PFLANZEBACK :LANG4 :HOCH4 :STEP4 :SUBL4 :SUBH4
setpencolor :FAR5 rt :WIN5
PFLANZEBACK :LANG5 :HOCH5 :STEP5 :SUBL5 :SUBH5
end
```

Kapitel 10

Aufgabe 11.2

```
to KLASSE2 :WAS :UM :GR
if :WAS=0 [KREISE :UM/360]
if :WAS=1 [fd :GR]
if :WAS=2 [TREPPE :GR :UM]
if :WAS>2 [VIELECK :WAS :GR]
if :WAS<0 [pr [Achtung, falsche Nummer]]
end
```

Aufgabe 11.3 Vier mögliche Aufrufe zum Testen der vier Fälle wären:

```
QUADMETH 0 10 5
QUADMETH 2 16 10
QUADMETH 2 4 2
QUADMETH 2 40 4
```

Wenn der Befehl `stop` entfernt wird, dann wird weitergearbeitet, obwohl `A` gleich Null ist. Dadurch entsteht ein Fehler, da später im Programm durch 2`A` und somit durch Null geteilt wird.

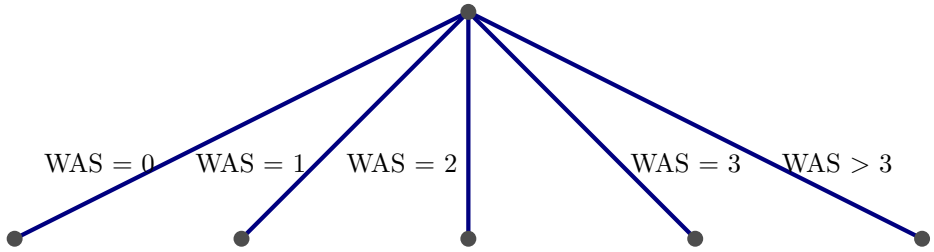
Aufgabe 11.4 Damit die Lösungen als Kreis dargestellt werden, muss nur der Aufruf `QUADRAT` durch den Aufruf `KREISE` ausgetauscht werden und der Parameter für `KREISE` entsprechend angepasst werden.

```
to QUADMETHKREIS :A :B :C
if :A=0 [ pr [keine quadratische Gleichung]stop]
make "M :B*:B-4*:A*:C pr :M
if :M<0 [ pr [keine reelle Loesung]]
if :M=0 [make "X0 (-:B)/(2*:A)
         pr [X0=] pr :X0 KREISE 100*:X0/360]
if :M>0 [make "X1 ((-:B))+sqrt :M/(2*:A)
         make "X2 ((-:B))-sqrt :M/(2*:A)
         pr [X1=] pr :X1 KREISE 100*:X1/360
         pr [X2=] pr :X2 KREISE 100*:X2/360]
end
```

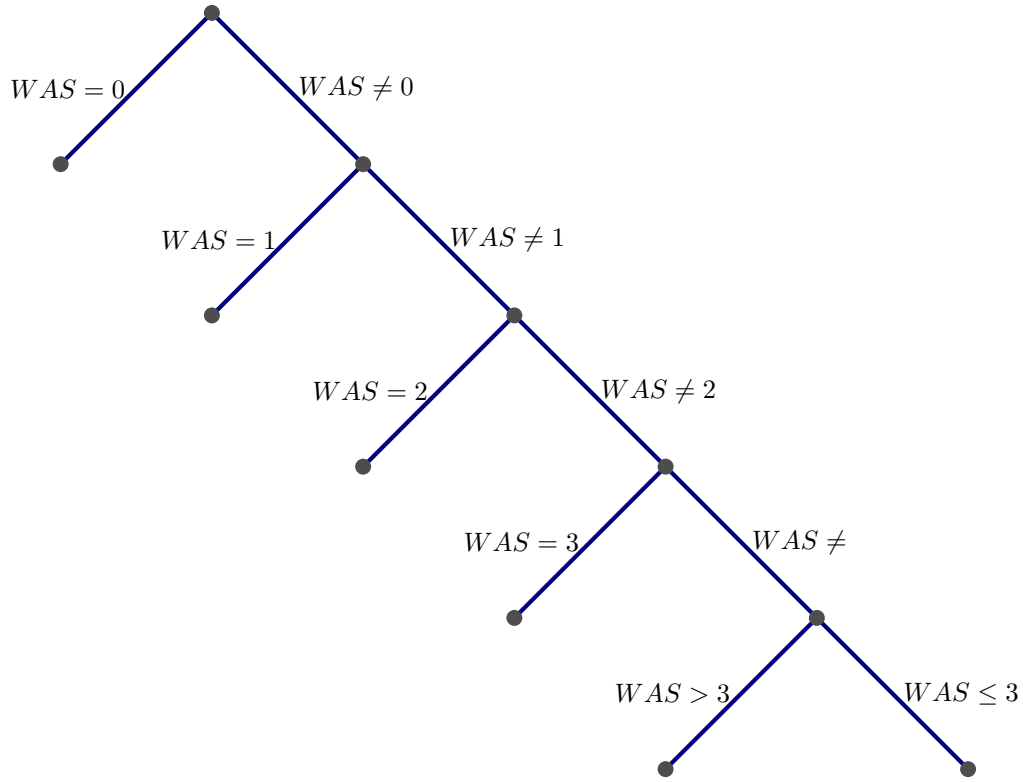
Aufgabe 11.5

```
to LINGLEICHUNG2 :A :B :C
if :A=0 [if :B =:C [pr [unendlich viele Loesungen] stop]
              pr [keine Loesung] stop]
make "X (:C-:B)/:A
pr [X=] pr :X QUADRAT 10*:X
end
```

Aufgabe 11.7 Das unveränderte Programm hat folgende Struktur:



Das Programm mit integrierten `stop`-Befehlen hat folgende Struktur:



Aufgabe 11.9

```
to LINGL2 :A :B :C :D
if :A=:C [if :B=:D [pr [alle reellen Zahlen] stop]]
if :A=:C [pr [keine Loesung] stop]
make "X (:D-:B)/(:A-:C)
pr [X=]
if :X>0 [LEITER :X]
if :X<0 [lt 180 LEITER (-:X)]
if :X=0 [KREISE 10/36]
end
```

Da der `repeat`-Befehl nicht mit negativen Zahlen umgehen kann, muss die Variable `X` mit -1 multipliziert werden, falls ihr Wert kleiner als Null ist. Leider wird hier das Bild nur gezeichnet, falls `X` eine ganze Zahl ist.

Aufgabe 11.10

```
to VIELECK2 :ECK :GR
repeat :ECK [fd :GR] rt 360/:ECK
end

to SECHSODERACHT :WAS
if :WAS=0 [VIELECK2 6 100] [VIELECK2 8 50]
end
```

Wenn der Parameter WAS genau Null beträgt, dann wird ein Sechseck gezeichnet. In allen anderen Fällen wird ein Achteck gezeichnet.

Aufgabe 11.12

```
to KLASSE2 :WAS :GR
if :WAS=0 [setpc 2 fd :GR]
  [if :WAS=1 [setpc 3 QUADRAT :GR/4]
     [if :WAS=2 [setpc 1 KREISE :GR/360]
        [if :WAS=3 [setpc 13 SCHW100]
            [if :WAS>3 [pr [Sorry, falsche Nummer]]]]]]
end
```

Aufgabe 11.13 Bei solch stark verschachtelten Programmen sollte stets darauf geachtet werden, dass die schliessenden Klammern in der richtigen Zahl am richtigen Ort stehen.

```
to QUADMETH3 :A :B :C
if :A=0 [ pr [keine quadratische Gleichung]]
    [make "M :B*:B-4*:A*:C pr :M
     if :M<0 [ pr [keine reelle Loesung]]
          [if :M=0 [make "X0 (-:B)/(2*:A)
                   pr [X0=] pr :X0 QUADRAT 10*:X0]
                        [if :M>0 [make "X1 ((-:B))+sqrt :M/(2*:A)
                                  make "X ((-:B))-sqrt :M/(2*:A)
                                  pr [X1=] pr :X1 QUADRAT 10*:X1
                                  pr [X2=] pr :X2 QUADRAT 10*:X2]]]]
end
```

Die Verzweigungsstruktur des Programmes QUADMETH3 ist identisch mit der Struktur von QUADMETH1.

Aufgabe 11.14

```
to LINGLEICHUNG3 :A :B :C
if :A=0 [if :B=:C [pr [unendlich viele Loesungen]] [pr [keine Loesung]]]
        [make "X (:C-:B)/:A pr [X=] pr :X QUADRAT 10*:X ]
end

to LINGL3 :A :B :C :D
if :A=:C [if :B=:D [pr [alle reelen Zahlen]] [pr [keine Loesung]]]
          [make "X (:D-:B) /(:A-:C) pr [:X=]
           if :X>0 [LEITER :X]
           if :X<0 [lt 180 LEITER (-:X)]
           if :X=0 [KREISE 10/36]]
end
```

Aufgabe 11.15 Jede Verzweigung der **??** ist als ein `if`-Befehl zu verstehen. Es werden also drei solcher Befehle verwendet. Da die zweite und dritte Verzweigung an den Ästen der ersten sind, müssen diese zwei in der ersten verschachtelt sein.

```
to KREISMAX :A :B :C
if  :A>:B [if :A>:C [KREISE :A/360] [KREISE :C/360]]
          [if :C>:B [KREISE :C/360] [KREISE :B/360]]
end
```

Aufgabe 11.16

```
to SPIRECK :LA :SUB :ECK
repeat 100 [fd :LA rt 360 /:ECK
            make "LA :LA-:SUB pr :LA
            if :LA-:SUB<0 [pr [LA zu klein]stop]]
end
```

Aufgabe 11.17 Diese Aufgabe ähnelt dem Programm `SPIRBED`. Ein Unterschied ist jedoch, dass der `if`-Befehl vor dem `make`-Befehl steht, da das Programm abgebrochen werden soll, wenn die letzte Linie die Länge 300 hat und nicht wenn die nächste Linie diese Länge erreicht hätte. Ausserdem sieht man an diesem Programmbeispiel, dass der Befehl `repeat` auch als eine Art Bedingung angesehen werden kann.

```
to SPIRIF :LA :ADD
repeat 200 [fd :LA rt 60
            if :LA+:ADD>300 [pr [LA zu gross] stop]
            make "LA :LA+:ADD pr :LA]
end
```

Aufgabe 11.18 Die Bedingung, dass nur zehn Halbkreise gezeichnet werden sollen, wird mit einer `repeat`-Schleife geregelt. Damit die Länge von `LA` nicht unter 50 fällt, wird am Ende der Schleife ein `if`-Befehl eingebaut, kombiniert mit dem Befehl `stop`.

```
to HALBKREISEIF :LA :RED
repeat 10 [repeat 180 [fd :LA/360 rt 1] pu
           repeat 180 [fd :LA/360 rt 1] pd
           make "LA :LA/:RED
           if :LA<50 [pr [LA zu klein]stop]]
end
```

Aufgabe 11.19 Die einzige Änderung, die an `PFLANZE` vorgenommen werden muss, ist der Einbau der Befehles `if` mit der entsprechenden Abbruchbedingung und dem Befehl `stop`

```
to PFLANZEIF :LANG :HOCH :STEP :SUBL :SUBH
repeat :HOCH [fd :STEP NADEL :LANG rt 3
              make "LANG :LANG-:SUBL
              if :LANG<0 [ pr [LANG zu klein]stop]
              make "STEP :STEP-:SUBH]
end
```

Aufgabe 11.20 Um diese Aufgabe zu lösen wird einfach eine zweite Bedingung in `HALBKREISEIF` eingfügt, die kontrolliert, ob `LA` nicht grösser als 1000 ist.

```
to HALBKREISEIF2 :LA :RED
repeat 10 [repeat 180 [fd :LA/360 rt 1] pu
           repeat 180 [fd :LA/360 rt 1] pd
           make "LA :LA/:RED
           if :LA<50 [pr [LA zu klein]stop]
           if :LA>1000 [pr [LA zu gross]stop]]
end
```

Aufgabe 11.23 Das Programm für die Spirale wird in eine `while`-Schleife gepackt, die die Ausführung des Programmes beendet, sobald `GR` den Wert 300 überschreitet.

```
to QUADSPIR :GR :ST
while [:GR< 300] [repeat 2 [fd :GR rt 90] make "GR :GR+:ST]
end
```

Aufgabe 11.24 Für diese Aufgabe wird eine `while`-Schleife erstellt, die das Programm zum Zeichnen von Quadraten so lange ausführt, bis `GR` unter den Wert 1 gesunken ist. In der `while`-Schleife wird `GR` nach jedem Durchlauf halbiert.

```
to QUADRATEWHILE
make "GR 200
while [:GR>1] [QUADRAT :GR
               make "GR :GR/2
               pu lt 90 fd :GR+3 rt 90 pd]
end
```

Aufgabe 11.25

```
to PFLANZEWHILE :LANG :STEP :SUBL :SUBH
while [:LANG>10] [fd :STEP NADEL :LANG rt 3
                  make "LANG :LANG-:SUBL
                  make "STEP :STEP-:SUBH]
end
```

Es gibt eine Möglichkeit, das Programm unendlich lange arbeiten zu lassen. Wenn der Parameter `SUBL` den Wert Null erhält, arbeitet das Programm unendlich lange, weil die Bedingung für die `while`-Schleife immer erfüllt ist.

Aufgabe 11.27 Wenn die Bedingung `:N>1` durch `:N>2` ersetzt wird, ändert sich nur der erste Wert, der ausgegeben wird.

Der Wert von `FA` bleibt jedoch für gleiche Eingaben identisch. Dies ist so, weil die Schleife mit der ursprünglichen Bedingung bei einem Wert von 2 für `N` einen Weiteren Durchlauf startet, in der zwar `N` um 1 reduziert wird, jedoch wird `FA` nicht verändert, da nur mit 1 multipiziert wird.

Aufgabe 11.28 Bei `FAK` wird bei der grössten Zahl begonnen und bei `FAK1` bei der kleinsten. Ausserdem benötigt `FAK1` drei Variablen, `FAK` dagegen nur zwei.

Die folgende Tabelle zeigt die Entwicklung der Variablen von `FAK1` nach den Durchläufen der `while`-Schleife.

Schleifendurchläufe	0	1	2	3	4	5
N	6	6	6	6	6	6
M	1	2	3	4	5	6
FA	1	2	6	24	120	720

Kontrollaufgabe 2 Zur Lösung dieser Aufgabe genügt eine einfache `if`-Verzweigung und zwei Programmaufrufe.

```
to PFLANZEODERSCHACH :WAS
if :WAS=0 [PFLANZE 50 100 20 0.5 0.1] [ SCHACHBEL 40 8]
end
```

Kontrollaufgabe 3

```
to MINMAX :A :B :C :D
if :A>:B [if :A>:C [if :A>:D [make "MAX :A make "X1 :B
                              make "X2 :C make "X3 :D]
                             [make "MAX :D make "X1 :B
                              make "X2 :C make "X3 :A]]
                   [if :C>:D [make "MAX :C make "X1 :B
                              make "X2 :A make "X3 :D]
                             [make "MAX :D make "X1 :B
                              make "X2 :C make "X3 :A]]]
          [if :C>:B [if :C>:D [make "MAX :C make "X1 :B
                              make "X2 :A make "X3 :D]
                             [make "MAX :D make "X1 :B
                              make "X2 :C make "X3 :A]]
                   [if :D>:B [make "MAX :D make "X1 :B
                              make "X2 :C make "X3 :A]
                             [make "MAX :B make "X1 :A
                              make "X2 :C make "X3 :D]]]
if :X1<:X2  if :X1<:X3 [make "MIN :X1][make "MIN :X3]]
           [if :X2<:X3 [make "MIN :X2][make "MIN :X3]]
pr [Maximum:] pr :MAX
pr [Minimum:] pr :MIN
end
```

Kontrollaufgabe 4 Es ist wichtig darauf zu achten, die Bedingung der `while`-Schleife zu überprüfen. Es ist schnell passiert, dass dort ein falscher Wert eingetragen wird und damit ein Schleifendurchlauf zu viel gemacht wird.

```
to PROG1 :AN
while [:AN>0] [P1 make "AN :AN-1]
end
```

Kontrollaufgabe 5 Einmal werden die Kreise in einer `while`-Schleife ausgeführt, die überprüft, ob der Umfang über 1000 ist und beim zweiten Programm wird die Grösse des Umfangs mit dem Befehl `if` kontrolliert. Um genügend Wiederholungen zu garantieren, wird das zweite Programm in einer `repeat`-Schleife mit sehr grossem Wert ausgeführt.

```
to KREISEWHILE :FAK
make "UM 70
if :FAK>1 [while [:UM<1000][repeat 360 [fd :UM/360 rt 1]
                            make "UM :UM*:FAK]
end
```

Die `if`-Verzweigung vor der `while`-Schleife ist notwendig, da das Programm sonst für Werte kleiner oder gleich 1 unendlich lange arbeiten würde.

```
to KREISEIF :FAK
make "UM 70
repeat 1000  [repeat 360 [fd :UM/360 rt 1] make "UM :UM*:FAK
                          if :UM>1000 [stop]]
end
```

Kontrollaufgabe 6 Zuerst werden die Stufen von unten links bis zur Spitze gezeichnet. Danach wird die Schildkröte um 90 Grad gedreht und dann zeichnet sie die Stufen von der Spitze nach unten rechts.

Als Erstes wird der Wert von GR in einer zweiten Variablen A gespeichert, damit später GR wieder bis auf ihren ursprünglichen Wert erhöht werden kann. Danach wird eine while-Schleife gestartet, in der solange Stufen gezeichnet werden, bis GR kleiner als 25 ist. Nach einer Drehung um 90 Grad beginnt eine zweite while-Schleife die solange Stufen zeichnet, bis GR wieder ihren ursprünglichen Wert besitzt.

```
to PYRWHILE :GR
make "A :GR
while [:GR>25] [ fd :GR rt 90 fd :GR lt 90 make "GR :GR-20]
rt 90
while [:GR<:A] [make "GR :GR+20 fd :GR rt 90 fd :GR lt 90]
end
```

Kontrollaufgabe 7 Bei diesem Programm sollte darauf geachtet werden, dass überprüft wird, ob ADD auch tatsächlich positiv ist.

```
to QUADWHILE :GR :ADD
if :ADD>0 [while [:GR<250]
                 [QUADRAT :GR make "GR :GR+:ADD]]
end
```

Kontrollaufgabe 8 Das Programm FUN2 führt die Funktion $f(n) = 2^n$ aus.

```
to FUN2 :N
make "F1 1
repeat :N [make "F1 :F1*2]
pr [n=] pr :N pr [f (n)=] pr :F1
end
```

Kontrollaufgabe 9 Es müssen drei Änderungen am Originalprogramm vorgenommen werden. Erstens muss die repeat-Schleife durch eine while-Schleife mit der Bedingung :UM<:MAX ersetzt werden, zweitens wird der Aufruf KREISE :UM/360 durch QUADRAT :UM/4 ersetzt und drittens wird AN durch MAX ausgetauscht.

```
to SPIRWHILE :UM :ADD :MAX
while [:UM<:MAX][QUADRAT :UM/4 fd :UM/2 rt 20
                 make "UM :UM+:ADD
                 make "ADD :ADD+10]
end
```

Kontrollaufgabe 10 Die Ausführung des Programmes wird nun durch eine while-Schleife gesteuert. Das Programm wird so lange ausgeführt, bis die Schildkröte im gesamten Programm mehr als ein Drehung um 90 Grad gemacht hat. Damit dies kontrolliert werden kann, wird eine neue Variable N erschaffen, die den gesamten Winkel aller gemachten Drehungen speichert. Da diese Variable in der Bedingung der while-Schleife enthalten ist, führt ein Gesamtwinke der mehr als 90 Grad beträgt zum Abbruch des Programmes.

```
to PFLANZEWHILE :LANG :STEP :SUBL :SUBH :WIN
make "N 0
while [:N<90] [fd :STEP NADEL :LANG rt :WIN
               make "LANG :LANG-:SUBL
               make "STEP :STEP-:SUBH
               make "N :N+:WIN]
end
```

Kapitel 11

Aufgabe 12.1 Das Programm zeichnet zwei Kreise mit Hilfe des Unterprogrammes KREISMITT. Zwischen den zwei Aufrufen wird die Schildkröte richtige positioniert.

```
to ZWEIKREISE :R :M
KREISMITT :R
pu rt 90 fd :M lt 90 pd
KREISMITT :R
end
```

Es gibt vier mögliche Schnittmengen. Für M>R ist die Schnittmenge leer, für M=R gibt es eine Schnittmenge mit genau einem Element, für M<R und M>0 gibt es in der Schnittmenge zwei Elemente und für M=R gibt es unendlich viele Elemente in der Schnittmenge. Vorraussetzung ist natürlich, dass R ungleich Null ist.

Aufgabe 12.2

```
to KREISMITT2 :R
pu lt 90 fd  :R rt 90 pd
KREISRAD :R
pu rt 90 fd :R lt 90 pd
end

to OLYMPRINGE :R
setpencolor 12
KREISMITT2 :R
pu rt 90 fd :R*2.2 lt 90 pd
setpencolor 0
KREISMITT2 :R
pu rt 90 fd :R*2.2 lt 90 pd
setpencolor 1
KREISMITT2 :R
pu lt 90 fd :R*1.1 rt 90 bk :R*1.1 pd
setpencolor 3
KREISMITT2 :R
pu lt 90 fd :R*2.2 rt 90 pd
setpencolor 11
KREISMITT2 :R
end
```

Aufgabe 12.3 Wenn mehrere Eingaben für KONSRECHTTR Null sind, dann wird, je nachdem welche Variablen betroffen sind, nur eine Strecke gezeichnet, eine Fehlermeldung ausgegeben weil die Zahl unter der Wurzel negativ ist oder gar nichts gezeichnet und keine Fehlermeldung ausgegeben. Wenn a negativ ist,dann wird das Dreieck nach links anstatt nach rechts gezeichnet. Wenn b negativ ist, dann wird das Dreieck nach unten gezeichnet. Wenn c negativ ist, dann wird dasselbe Dreieck gezeichnet, wie wenn c positiv wäre. Falls a oder b grösser sind als c, dann wird eine Fehlermeldung ausgegeben.

Im folgenden Programm werden die Eingaben überprüft und es wird eine Warnung ausgegeben, falls eine der Eingaben ungültig ist.

```
to KONSRECHTTR2 :a :b :c
if :a<0 [pr [Eine Eingabe ist negativ!]stop]
if :b<0 [pr [Eine Eingabe ist negativ!]stop]
if :c<0 [pr [Eine Eingabe ist negativ!]stop]
if :c>0 [if :a>:c pr [c muss den groessten Wert haben oder
                     Null sein]stop]
        [if :b>:c pr [c muss den groessten Wert haben oder
                     Null sein]stop]]
if :a=0 [if :b=0 [pr [Es ist mehr als eine Eingabe Null!]stop]
         if :c=0 [pr [Es ist mehr als eine Eingabe Null!]stop]
         make "a sqrt (:c*:c-:b*:b) DREI90 :a :b
         pr :a pr :b pr :c home stop]
if :b=0 [if :a=0 [pr [Es ist mehr als eine Eingabe Null!]stop]
         if :c=0 [pr [Es ist mehr als eine Eingabe Null!]stop]
         make "b sqrt (:c*:c-:a*:a)
         DREI90 :a :b
         pr :a pr :b pr :c home stop]
if :c=0 [if :a=0 [pr [Es ist mehr als eine Eingabe Null!]stop]
         if :b=0 [pr [Es ist mehr als eine Eingabe Null!]stop]
         make "c sqrt (:a*:a+:b*:b)
         DREI90 :a :b
         pr :a pr :b pr :c home stop]
pr [Genau eine der Eingaben muss Null sein!]
end
```

Das Programm KONSRECHTTR kann gekürzt werden, indem alle Befehle stop und home weggelassen werden. Falls die Eingaben korrekt sind, arbeitet das Programm auch ohne diese Befehle korrekt.

```
to KONSRECHTTRKURZ :a :b :c
if :a=0 [make "a sqrt (:c*:c-:b*:b)
      DREI90 :a :b
      pr :a pr :b pr :c]
if :b=0 [make "b sqrt (:c*:c-:a*:a)
      DREI90 :a :b
      pr :a pr :b pr :c]
if :c=0 [make "c sqrt (:a*:a+:b*:b)
      DREI90 :a :b
      pr :a pr :b pr :c]
end
```

Aufgabe 12.5

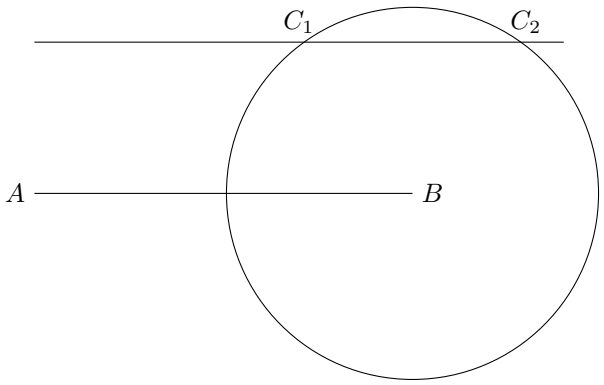
```
to DREIWSW :w1 :a :w2
if (:w1+:w2)>180 [pr [Zu grosse Winkel] stop]
rt 90-:w1 fd 1000 bk 1000
rt :w1 fd :a lt 180-:w2 fd 1000 bk 1000
end
```

Aufgabe 12.10 Zuerst wird ein Kreis mit dem Radius a gezeichnet. Vom Kreismittelpunkt aus wird die Seite c gezeichnet, parallel zu c mit einer Distanz von h_c wird eine sehr lange Strecke gezeichnet. Nun sind die drei Eckpunkte des Dreiecks definiert. Folgendes Programm zeichnet genau diese Abbildung:

```
to PARALLEL2 :DIST :LA
rt 90 fd :LA/2 bk :LA fd :LA/2
lt 90 pu fd :DIST pd
rt 90 fd 500 bk 1000 fd 500
end

to DREIACHC :a :c :hc
if :a<0 [pr [Die Eingaben muessen alle positiv sein]stop]
if :c<0 [pr [Die Eingaben muessen  alle positiv sein]stop]
if :hc<0 [pr [Die Eingaben muessen  alle positiv sein]stop]
if :a<:hc [pr [hc darf nicht groesser sein als a]stop]
KREISMITT :a
lt 90 fd :c/2 rt 90
PARALLEL2 :hc :c
```

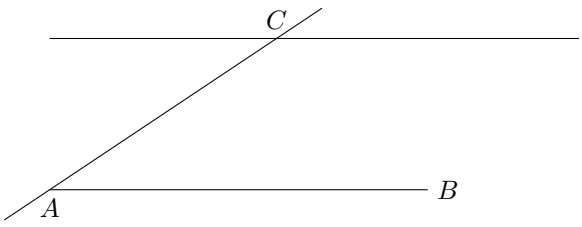
Die `if`-Befehle sind nötig, um die Eingabewerte auf ihre Gültigkeit zu prüfen. Die Konsturktion sieht dann so aus:



Natürlich gibt es noch zwei weitere Punkte, die mit einer Modifikation des Programmes ebenfalls eingezeichnet werden können.

Aufgabe 12.11 Zuerst wird eine Gerade gezeichnet, auf der die Strecke b liegt. Um diese Grade zu zeichnen wird die Schildkröte um den Winkel $90 - \alpha$ nach rechts gedreht, dann kann gezeichnet werden. Im zweiten Schritt wird die Strecke c und eine Parallele im Abstand h_c gezeichnet.

Dazu wird die Schildkröte um den Winkel α gedreht, damit sie waagrecht steht. Nun erkennt man den Punkt A am linken Ende von c, B am rechten Ende von$c$ und C im Schnittpunkt der Strecke a mit der Parallelen zu c. Die Abbildung dazu sieht so aus:



Das Programmdazu sieht so aus:

```
to DREICALPHAHC :c :alpha :hc
if :c<0 [pr [Die Eintraege muessen positiv sein!]stop]
if :hc<0 [pr [Die Eintraege muessen positiv sein!]stop]
```

```
if :alpha<0 [pr [Die Eintraege muessen positiv sein!]stop]
if :alpha>180 [pr [alpha kann nicht groesser als 180 Grad sein]stop]
rt 90-:alpha fd 1000 bk 1000 rt :alpha fd :c/2 lt 90
PARALLEL2 :hc :c
end
```

Aufgabe 12.12 Bevor programmiert wird, müssen die Gleichungen nach x und nach y aufgelöst werden. Eine Möglichkeit besteht darin, beide Gleichungen nach y aufzulösen und dann gleichzusetzen. Die erste Gleichung wird nach y aufgelöst:

$$\begin{aligned} ax + by &= c \\ by &= c - ax \\ y &= \frac{c - ax}{b} \end{aligned}$$

Die zweite Gleichung wird ebenfalls aufgelöst:

$$\begin{aligned} dx + ey &= f \\ ey &= f - dx \\ y &= \frac{f - dx}{e} \end{aligned}$$

Nun können diese Gleichungen gleichgestellt werden und nach x aufgelöst werden:

$$\begin{aligned} \frac{c - ax}{b} &= \frac{f - dx}{e} \\ ce - aex &= bf - bdx \\ bdx - aex &= bf - ce \\ x(bd - ae) &= bf - ce \\ x &= \frac{bf - ce}{bd - ae} \\ x &= \frac{ce - bf}{ae - bd} \end{aligned}$$

Nun kann x in die folgende Gleichung eingesetzt werden:

$$\begin{aligned}
y &= \frac{f - dx}{e} \\
&= \frac{f - d\frac{bf-ce}{bd-ae}}{e} \\
&= \frac{f}{e} - \frac{d(bf-ce)}{e(bd-ae)} \\
&= \frac{f(bd-ae) - d(bf-ce)}{e(bd-ae)} \\
&= \frac{bdf - aef - bdf + cde}{e(bd-ac)} \\
&= \frac{-aef + cde}{e(bd-ac)} \\
&= -\frac{e(af-cd)}{(bd-ae)} \\
y &= \frac{af-cd}{ae-bd}
\end{aligned}$$

Nun kann mit dem Programmiern begonnen werden. Das Programm kann folgendermassen aussehen:

```
to ZWEILINGL :a :b :c :d :e :f
if (:a*:e-:b*:d)=0 [pr [Gleichung nicht loesbar] stop]
make "X (:e*:c-:b*:f)/(:a*:e-:b*:d)
pr [X=] pr :X
make "Y (:f*:a-:d*:c)/(:a*:e-:b*:d)
pr [Y=] pr :Y
end
```

Aufgabe 12.13 Um herauszufinden, wie das Verhältnis von Höhe (a) zu Breite (b) sein muss, damit das Rechteck die maximale Fläche hat, muss eine Funktion der Fläche erstellt werden. Dies geschieht mit Hilfe der Formel für die Fläche (A) eines Rechteckes und der Formel für den Umfang(U):

$$A = a \cdot b$$

$$\begin{aligned}
U &= 2a + 2b \\
2a &= U - 2b \\
a &= \frac{U-2b}{2}
\end{aligned}$$

Wenn nun die beiden Formeln kombiniert werden, dann erhält man die gewünschte Funktion:

$$\begin{aligned}
A(b) &= \frac{U-2b}{2} \cdot b \\
A(b) &= \frac{Ub-2b^2}{2}
\end{aligned}$$

Damit die Extremalwerte dieser Funktion ermittelt werde können, muss diese Funktion abgeleitet werden:

$$A'(b) = \frac{U-4b}{2}$$

Ist die Ableitung gleich Null gesetzt, können die Extremalwerte ermittelt werden:

$$\frac{U-4b}{2} = 0$$
$$U - 4b = 0$$
$$U = 4b$$

Der Umfang eines Rechtecks mit maximaler Fläche beträgt also $4b$. Dies ist jedoch nur der Fall, wenn alle vier Seiten des Rechtecks gleich lang sind. Also ist ein Rechteck mit konstantem Umfang und maximaler Fläche ein Quadrat. Das Programm dazu lautet natürlich:

```
to RECHTMAX :UM
QUADRAT :UM/4
end
```

Aufgabe 12.14 Nein

Aufgabe 12.15 Die `while`-Schlaufe ist dafür zuständig zu kontrollieren, dass die Zahl gefunden wird, die die Bedingungen der Aufgabe erfüllen. Nachdem die richtige Zahl ermittelt worden ist, wird zum grössten Teil derselbe Code ausgeführt wie im **??**, nur die Farbe wird nach jedem Würfel noch gewechselt, sowie die richtige Variable bereitgestellt.

```
to VIERWURFEL :M
make "A 1
while [(:A*:A*:A+(:A+50)*(:A+50)*(:A+50)+
       (:A+100)*(:A+100)*(:A+100)+(:A+150)*(:A+150)*(:A+150))<:M]
      [make "A :A+1]
setpencolor red
pr :A QUADRAT :A
rt 45 fd :A/2 lt 45 QUADRAT :A rt 45 bk :A/2 lt 45
fd :A rt 45 fd :A/2 bk :A/2 rt 45
fd :A lt 45 fd :A/2 bk :A/2 rt 135
fd :A lt 135 fd :A/2 bk :A/2 lt 45
make "A :A+50
setpencolor yellow
pr :A QUADRAT :A
rt 45 fd :A/2 lt 45 QUADRAT :A rt 45 bk :A/2 lt 45
fd :A rt 45 fd :A/2 bk :A/2 rt 45
fd :A lt 45 fd :A/2 bk :A/2 rt 135
fd :A lt 135 fd :A/2 bk :A/2 lt 45
make "A :A+50
setpencolor blue
pr :A QUADRAT :A
rt 45 fd :A/2 lt 45 QUADRAT :A rt 45 bk :A/2 lt 45
fd :A rt 45 fd :A/2 bk :A/2 rt 45
fd :A lt 45 fd :A/2 bk :A/2 rt 135
fd :A lt 135 fd :A/2 bk :A/2 lt 45
make "A :A+50
setpencolor green
pr :A QUADRAT :A
rt 45 fd :A/2 lt 45 QUADRAT :A rt 45 bk :A/2 lt 45
```

```
fd :A rt 45 fd :A/2 bk :A/2 rt 45
fd :A lt 45 fd :A/2 bk :A/2 rt 135
fd :A lt 135 fd :A/2 bk :A/2 lt 45
end
```

Aufgabe 12.17

```
to NULLSTELLE1000 :a :b :c :d
make "m :a+:b+:c+:d make "X 1 make "Y 2
if :m=0 [pr :X stop]
if :m>0 [while [:a*:Y*:Y+:b*:Y*:Y+:c*:Y+:d>0]
                          [make "X :X+1 make "Y :Y+1
                           if :X >1000 [pr [x groesser als 1000] stop]]]
if :m>0 [while [:a*:Y*:Y+:b*:Y*:Y+:c*:Y+:d<0]
                          [make "X :X+1 make "Y :Y+1
                           if :X >1000 [pr [x groesser als 1000] stop]]]
pr :X
end
```

Dieses Programm ist eine um zwei `if`-Befehle erweiterte Version des Programmes `NULLSTELLEN` aus dem Buch auf Seite 211. Diese beiden hinzugefügten Befehle sollen kontrollieren, ob `X` grösser als 100 ist, und wenn dies der Fall ist, das Programm abbrechen.

Kontrollaufgabe 1 Die Zahl R, die als Radius verwendet werden soll, ist die kleinste Zahl, die diese Ungleichung erfüllt:

$$\begin{aligned} R + (R+2) + (R+4) + (R+6) &> M \\ 4R + 12 &> M \end{aligned}$$

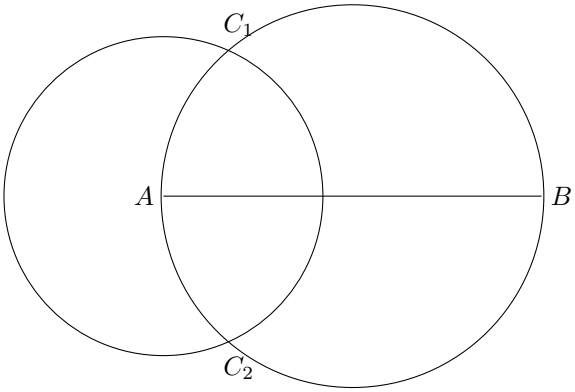
Um dies zu erreichen, kann im Programm eine `while`-Schleife verwendet werden. Danach werden ebenfalls mit einer `while`-Schleife die Halbkreise gezeichnet.

```
to HALBKREISRADR :R
make "UM 2*3.1415927*:R
repeat 180 [fd :UM/360 rt 1]
end

to HALBKREISRADL :R
make "UM 2*3.1415927*:R
repeat 180 [fd :UM/360 lt 1]
end

to WELLE :M
make "R 0
while [4*:R+12 <:M] [make "R :R+1]
pr :R
while [:R>10] [HALBKREISRADR :R  make "R :R-10
               if :R<10 [stop]
               HALBKREISRADL :R  make "R :R-10]
end
```

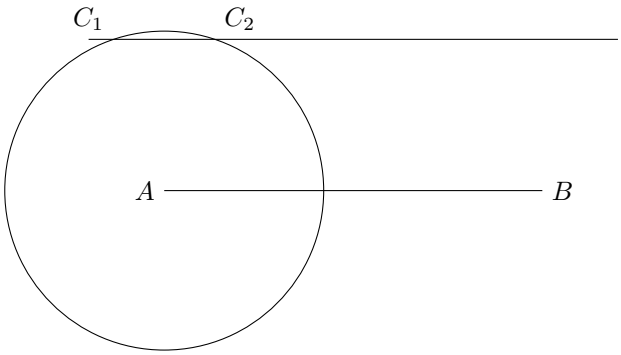
Kontrollaufgabe 2 Das rechtwinklige Dreieck kann mit Hilfe des Thaleskreises aus den beiden Seiten b und c konstruiert werden. Zuerst wird die Hypothenuse c und der Thaleskreis eingezeichnet. Danach kann von der Ecke A aus ein Kreis mit dem Radius b gezeichnet werden. Somit ist die Konstruktion schon beendet und sollte etwa so aussehen:



Als Programm sieht die Kontsruktion so aus:

```
to DREIRECHTBC :b :c
KREISMITT :c/2
rt 90 fd :c/2 bk :c lt 90
KREISMITT :b
end
```

Kontrollaufgabe 3 Die Konstruktion beginnt mit einem Kreis mit Radius a. Danach wird die Seite b eingezeichnet, sowie eine Parallelle dazu im Abstand h_c Das Ganze sieht dann etwa so aus:



Nun kann das Programm geschreiben werden.

```
to PARALLEL2 :DIST :LA
rt 90 fd :LA/2 bk :LA fd :LA/2
lt 90 pu fd :DIST pd
rt 90 fd 500 bk 1000 fd 500
end

to DREIABHB :a :b :hb
KREISMITT :a
rt 90 fd :b/2 lt 90
PARALLEL2 :hb :b
end
```

Kontrollaufgabe 4

```
to PARA :a :b :alpha
if :alpha> 90 [pr [alpha zu gross [stop]]
if :alpha< 0 [pr [alpha muss positiv sein] stop]
rt 90-:alpha
fd :b rt :alpha fd :a
rt 180-:alpha
fd :b rt :alpha fd :a
end
```

Kontrollaufgabe 5 Die Koordinate x lässt sich ermitteln, indem die beiden Funktionen einader gleichgesetzt werden. Daraus ergibt sich eine quadratische Gleichung, die es in Normalform zu bringen gilt. Hat man die Normalform, kann die Lösungsformel angewandt werden.

$$\begin{aligned} ax^2 + b &= cx + d \\ ax^2 - cx + b - d &= 0 \\ x &= \frac{c \pm \sqrt{c^2 - 4ac(b-d)}}{2a} \end{aligned}$$

Die Diskriminante verrät die Anzahl der Schnittpunkte der beiden Funktionen. Ist die Diskriminante kleiner als Null gibt es keine Schnittpunkte, ist sie gleich Null gibt es genau einen Schnittpunkt und für einen Wert grösser als Null gibt es zwei Schnittpunkte. Die Koordinate in y-Richtung lässt sich ermitteln, indem man x in eine der beiden Funktionen einsetzt.

```
to KOORDINATEN :a :b :c :d
make "disk :c*:c-4*:a*(:b-:d)
if :disk<0 [pr [keine Schnittpunkte]stop]
if :disk=0 [make "X :c/(2*:a) make "Y :c*:X+:d
            pr [x=] :X pr [y=] pr :Y stop]
if :disk>0 [make "X1 (:c+sqrt (:disk))/(2*:a)
            make "X2 (:c-sqrt (:disk))/(2*:a)
            make "Y1 :c*:X1+:d make "Y2 :c*:X2+:d
            pr [x1=] pr :X1 pr [y1=] pr :Y1
            pr [x2=] pr :X2 pr [y2=] pr :Y2]
end
```

Kontrollaufgabe 6 Das x wird ermittelt, indem das Programm einen Wert für x einsetzt. Danach werden die Werte der beiden Funktionen berechnet und verglichen. Ist die quadratische Funktion nicht mehr kleiner als die lineare, wird die Suche abgebrochen. Es muss nur noch festgestellt werden, ob die Funtionen für das x, das zuletzt eingesetzt wurde gleich gross waren oder nicht. Waren sie gleich gross, dann ist x die Lösung. Falls die Quadratische jedoch grösser war, dann muss x um 1 verringert werden, da es sonst die Anforderungen nicht mehr erfüllt.

```
to FUNVERGLEICH :a :b :c :d
make "X 1
if (:a*:X*:X+:b>:c*:X+:d) [pr [Es gibt keine Loesung] stop]
while [:a*:X*:X+:b<:c*:X+:d][make :X :X+1]
if (:a*:X*:X+:b=:c*:X+:d) [pr [X=] pr :X]
                          [make :X-1 pr [X=] pr :X]
end
```

Kontrollaufgabe 7

```
to POLYVERGLEICH :GR
make "X 1
if (:GR<0) [pr [Eingabe muss groesser als Null sein]stop]
while [:X*:X*:X*:X-:X*:X*:X+:X*:X-:X<:GR][make "X :X+1]
if (:X*:X*:X*:X-:X*:X*:X+:X*:X-:X=:GR)[make "X :X+1]
pr [x=] pr :X
end
```

Kontrollaufgabe 8 Um zu überprüfen, ob `X` gerade ist, kann man eine `while`-Schleife benutzen, die `X` solange um 2 vermindert, bis `X` kleiner als 2 ist. Wenn der Wert der Variablen dann 1 beträgt, war `X` ungerade und für 0 war `X` gerade.

```
to MOD2 :X
if :X<0 [KREISE :X/360]
make "N :X
while [:N>1] [make "N :N-2]
if :N=0 [QUADRAT :X]
if :N=1 [KREISE :X/360]
end
```

Die Variable `N` ist nötig, damit der ursprüngliche Wert von `X` erhalten bleibt um die Figuren zu zeichnen.

Kontrollaufgabe 9 Je kleiner die Differenz zwischen den Seitenlängen, desto grosser ist die Fläche des Rechtecks bei konstantem Umfang. Ist der Umfang durch vier teilbar, dann sind die Seitenlängen gleich gross. Ist dies nicht der Fall, so hat eine Seite den Wert $(\texttt{UM}-2)/4$ und die andere $(\texttt{UM}+2)/4$.

```
to QUADMAX :UM
if :UM<4 [pr [UM ist zu klein]stop]
make "N :UM
while [:N>3] [make "N :N-4]
if :N=0 [make "a :UM/4 make "b :a
         pr [a=] pr :a pr [b=] pr :b stop]
if :N=2 [make "a (:UM-2)/4 make "b (:UM+2)/4
         pr [a=] pr :a pr [b=] pr :b stop]
pr [UM ist nicht gerade]
end
```

Zuerst wird geprüft, ob der Umfang genug gross ist. Falls er kleiner als 4 ist, wird das Programm abgebrochen, da für diese Werte keine Lösung existiert. Dann wird ermittelt, ob `UM` durch vier teilbar ist. Abhängig vom Ergebnis werden dann die Seitenlängen angegeben.

Kontrollaufgabe 10

```
to VERGLEICHX :X
if 2>(:X*2+1) [pr [keine Loesung]stop]
make "n 1
make "A 2
while [:A<(:X*2+1)] [make "A :A*2 make "n :n+1]
pr [n=] pr :n
end
```

In der ersten Zeile wird geprüft, ob der Wert von `X` nicht so klein ist, dass gar keine Lösungen existieren. Ist dies nicht der Fall, wird in einer `while`-Schleife die Grösse von n ermittelt. Da es in LOGO nur die vier Grundoperationen gibt, muss der Wert von n mit Hilfe der Multiplikation und eines Zählers ermittelt werden. Der Wert des Zählers entspricht genau n.

Kontrollaufgabe 11 Die Summe (s) von m und den $a \cdot m$ Nachfolgezahlen kann mit Hilfe der Gauss'schen Summenformel berechnet werden:

$$\begin{aligned}
s &= m + \frac{(m+am)(m+am+1)}{2} - \frac{m(m+1)}{2} \\
&= m + \frac{m^2 + am^2 + m + am^2 + a^2m^2 + am - m^2 - m}{2} \\
&= m + \frac{a^2m^2 + 2am^2 + am}{2} \\
&= \frac{2m + a^2m^2 + 2am^2 + am}{2} \\
&= \frac{m(2 + a^2m + 2am + a)}{2}
\end{aligned}$$

Das Programm rechent nun die Fakultät von m in einer `while`-Schleife aus und vergleicht diese mit der Summe s.

Danach wird m um 1 erhöht. Ist die Fakultät grösser oder gleich wie die Summe wird die `while`-Schleife beendet.

Nun muss noch geprüft werden, ob die Fakultät gleich gross oder grösser als die Summe ist. Ist die Fakultät gleich gross, dann muss m noch ein letztes Mal erhöht werden.

```
to FAKM :a
if :a<1 [pr [a muss groesser als Null sein] stop]
make "mfak 1 make "sum 2 make "m 1
while [:sum>:mfak] [make "m :m+1
                    make "mfak :mfak*:m
                    make "sum (:m*(2+:a*:a*:m+2*:a*:m+:a))/2]
pr [m=] pr :m
end
```

Kapitel 12

Aufgabe 13.1

```
to EWIG
                                EWIG
                                fd 100 rt 90
                        fd 100 rt 90
                fd 100 rt 90
        fd 100 rt 90
fd 100 rt 90
end
```

Aufgabe 13.2

```
to EWIG1
fd 100 rt 90 wait 1000
        fd 100 rt 90 wait 1000
                fd 100 rt 90 wait 1000
                        fd 100 rt 90 wait 1000
                        EWIG1
end
```

Aufgabe 13.3

```
to SPIRINF
fd 20 rt 90 wait 1000 pr [20]
make "LA 20+2
SPIRINF 22
        fd 20 rt 90 wait 1000 pr [22]
        make "LA 22+2
        SPIRINF 24
                fd 24 rt 90 wait 1000 pr [24]
                make "LA 24+2
                SPIRINF 26
                        fd 26 rt 90 wait 1000 pr [26]
                        make "LA 26+2
                        SPIRINF 28
                                fd 28 rt 90 wait 1000 pr [28]
                                make "LA 28+2
                                SPIRINF 30
                                end
                        end
                end
        end
end
```

Aufgabe 13.6

```
to SECHSINF :LA
fd :LA rt 60 wait 1000
SECHSINF :LA+2
end
```

Aufgabe 13.8

(a) Das Programm zeichnet dasselbe Bild mit und ohne `stop`.

(b) Das geänderte Programm zeichnet das gleiche Bild wie das Original, ausser `:MAX` ist ein Vielfaches von `LA`. In diesem Fall wird im geänderten Programm noch eine Linie gezeichnet, im Original dagegen nicht.

(c) Das geänderte Programm zeichnet eine Linie mehr als das Original, da `LA` erst nach dem Beginn des neuen Durchlaufes der `while`-Schleife erhöht wird.

Aufgabe 13.9 Das Programm zeichnet eine rechteckige Spirale.

```
to SPIRRECHTREC2 :MIN1 :MIN2 :ADD1 :ADD2 :MAX
if :MIN1>:MAX [stop]
          [if :MIN2>:MAX [stop]
                [fd :MIN1 rt 90 fd :MIN2 rt 90
                    make "MIN1 :MIN1+:ADD1 make "MIN2 :MIN2+:ADD2
                    SPIRRECHTREC2 :MIN1 :MIN2 :ADD1 :ADD2 :MAX]]
end
```

Aufgabe 13.12

```
to PYRREC :GR :N
FELDREC :GR :N
fd :GR lt 90 fd :GR*(:N-1) rt 90
make "N :N-2
if :N<1 [stop]
PYRREC :GR :N
end
```

Dieses rekursive Programm beinhaltet ein Unterprogramm, das ebenfalls rekursiv ist. Zuerst wird `FELDREC` aufgerufen und zeichnet eine Reihe mit `N` Quadraten. Danach wird die Schildkröte auf die Starposition für die nächste Zeile gesetzt und dann wird das Hauptprogramm mit einem kleineren `N` aufgerufen. Dies geschieht solange, bis `N` kleiner als 1 ist.

Aufgabe 13.13 Die `repeat`-Anweisung wird durch eine `while`-Schleife und den Befehl `make` ersetzt.

```
to AUGE2 :AN :UM :NACH
while [:AN>0] [KREISE :UM/360 make "UM :UM+:NACH
               make "AN :AN-1]
end
```

Um ein rekursives Programm zu erzeugen wird die `while`-Schleife durch einen `if`-Befehl und eine rekursiven Aufruf ersetzt.Der Befehl `if` dient zum Abbruch des Programmes sobald die Variable `AN` kleiner als 1 ist.

```
to AUGE3 :AN :UM :NACH
KREISE :UM/360
make "UM :UM+:NACH
make "AN :AN-1
if :AN<1 [stop]
AUGE3 :AN :UM :NACH
end
```

Aufgabe 13.14

```
to TREPPEREC :GR :AN
fd :GR rt 90 fd :GR lt 90
make "GR :GR+5
make "AN :AN-1
if :AN<1 [stop]
TREPPEREC :GR :AN
end
```

Aufgabe 13.15

```
to DREIREC :LA
fd :LA rt 90 fd :LA
make "LA sqrt (:LA*:LA*2)
rt 135 fd :LA rt 180
if :LA>400 [stop]
DREIREC :LA
end
```

Aufgabe 13.17

```
to FELDREC :GR :N
if :N=0 [stop]
make "N :N-1
repeat 7 [fd :GR rt 90] rt 90 wait 10
FELDREC :GR :N
pr :N
end
```

Aufgabe 13.18 Wenn der Befehl `make` nicht benutzt wird, aber stattdessen der Aufruf `FELDREC :GR :N-1` erfolgt, wird die Variable `N` des aufrufenden Programmes nicht geändert. `N(FELDREC)` bleibt also für den ganzen Programmablauf 4, ebenso bleibt `N(FELDREC 30 1)` immer 1.

Wenn die Zeile `make "GR 2*:GR make "N :N+7` eingefügt wird, dann ändern sich zwar vor dem Beenden jedes Aufrufes die Variablen, aber auf die Zeichnung hat dies keine Auswirkung.

Aufgabe 13.19

	0	1	2	3	4
MIN1(SPIRRECHTREC)	100	110	110	110	110
MIN2(SPIRRECHTREC)	10	15	15	15	15
ADD1(SPIRRECHTREC)	10	10	10	10	10
ADD2(SPIRRECHTREC)	5	5	5	5	5
MAX(SPIRRECHTREC)	120	120	120	120	120
MIN1(SPIRRECHTREC 110 15 10 5)	-	110	120	120	-
MIN2(SPIRRECHTREC 110 15 10 5)	-	15	20	20	-
ADD1(SPIRRECHTREC 110 15 10 5)	-	10	10	10	-
ADD2(SPIRRECHTREC 110 15 10 5)	-	5	5	5	-
MAX(SPIRRECHTREC 110 15 10 5)	-	120	120	120	-
MIN1(SPIRRECHTREC 120 15 10 5)	-	-	120	-	-
MIN2(SPIRRECHTREC 120 15 10 5)	-	-	20	-	-
ADD1(SPIRRECHTREC 120 15 10 5)	-	-	10	-	-
ADD2(SPIRRECHTREC 120 15 10 5)	-	-	5	-	-
MAX(SPIRRECHTREC 120 15 10 5)	-	-	120	-	-

Aufgabe 13.20 Das Programm macht zuerst alle Aufrufe, bis es die angegebene Tiefe erreicht hat. Erst in diesem Moment beginnt die Schildkröte zu zeichnen. Nach dem Zeichnen eines Kreises kann das Programm eine Rekursionsstufe höher mit dem Zeichnen eines Viertelkreises weiterfahren. Da die Variable TIEF dieses Aufrufes aber nicht Null ist, geht das Pogramm wieder eine Rekursionsstufe nach unten und zeichnet wieder einen der kleinsten Kreise. Ist auch der zweitkleinste Kreis vollständig gezeichnet, kann das Programm auf die drittunterste Rekursionsstufe springen. Dieses Vorgehen dauert so lange, bis der Kreis der obersten Stufe gezeichnet wurde.

Aufgabe 13.21

```
to KREISREC2 :UM :TIEF
if :TIEF=0 [stop]
repeat 3 [KREISREC2 :UM/3 :TIEF-1
                repeat 120 [fd :UM/360 rt 1] wait 200]

end
```

Die Änderungen betreffen nur die repeat-Schleifen. Die Anzahl Wiederholungen müssen von 90 auf 120 gesetzt werden, da nun Drittelkreise statt Viertelkreise gezeichnet werden sollen. Der Wert bei der zweiten Schleife muss von 4 auf 3 gesetzt werden, damit nur dreimal ein Drittelkreis gezeichnet wird.

Aufgabe 13.24

(	(	(	(	(	(	(	(	)	)	)	)	)	)	)	)
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

Aufgabe 13.25

$R_3 = (R_2)(R_2)(R_2)(R_2)$

Die Klammernfolge von KREISREC 1000 4 besteht aus vier Klammern mit dem Inhalt R_3.

Aufgabe 13.27 Die öffnenden Klammern stehen für begonnene Aufrufe, die schliessenden Klammern für beendete Aufrufe. Die Tiefe eines rekursiven Programmes ist die maximale Anzahl der in sich verschachtelten Aufrufe eines Programmes. Wenn man nun bis zu einem Punkt in der Klammernfolge alle öffnenden Klammern zählt und davon alle schliessenden Klammern abzieht, die dazwischen vorgekommen sind, dann hat man alle noch nicht beendeten, in sich verschatelten Aufrufe und somit die Rekursionstiefe dieses Punktes ermittelt. Nun muss nur noch der Punkt im Programm ermittelt werden, der die grösste Tiefe besitzt, um festzustellen, wie die Rekursionstiefe lautet.

Aufgabe 13.28 Damit sich die Kreise in der Mitte berühren, muss sich der Radius der Kreise beim rekursiven Aufruf halbieren. Deshalb wird dein neuer Aufruf nur mit der Hälfte des Radius des aufrufenden Programmes getätigt. Mit Hilfe des Radius kann nun vor dem Zeichnen der benötigte Umfang berechnet werden.

```
to KREISRADREC :R :TIEF
if :TIEF=0 [stop]
repeat 2 [KREISRADREC :R/2 :TIEF-1
          make "UM 2*3.1415927*:R
          repeat 180 [fd :UM/360 rt 1]]
end
```

Aufgabe 13.29

```
to KREUZREC :GR :TIEF
if :TIEF=0 [stop]
repeat 4 [KREUZREC :GR/3 :TIEF-1
                 fd :GR rt 90 ]
end
```

Aufgabe 13.31 Die Reihennfolge der Quadrate kann mittels Einbau des Befehls `wait` am laufenden Programm beobachtet werden.

Aufgabe 13.32

```
to TREPPEREC2 :LA :TIEF
if :TIEF=0 [fd :LA stop]
TREPPEREC2 :LA/3 :TIEF-1
lt 90
TREPPEREC2 :LA /3 :TIEF-1
rt 90
TREPPEREC2 :LA/3 :TIEF-1
rt 90
TREPPEREC2 :LA/3 :TIEF-1
lt 90
TREPPEREC2 :LA/3 :TIEF-1
end
```

Aufgabe 13.33

```
to DREIREC2 :LA :TIEF
if :TIEF=0 [stop]
DREIREC2 :LA/2 :TIEF-1
repeat 4 [fd :LA rt 120]
pu lt 60 fd :LA rt 120 pd
end
```

Aufgabe 13.34

(a) Das Baumdiagramm zum Aufruf `BAUM 80 4` ist analog zu der **??** im Buch. Der Unterschied liegt in der Anzahl Ebenen. Während **??** noch vier Ebenen hat, hat der Aufruf `BAUM 80 4` zusätzlich eine fünfte Ebene.

Die Klammernfolge sieht so aus:

$$R_1 = ()()$$
$$R_2 = (R_1)(R_1)$$
$$R_3 = (R_2)(R_2)$$
$$R_4 = (R_3)(R_3)$$

(b) Das Baumdiagramm zu `FELDABREC 200 2` hat drei Ebenen. Vom Hauptprogramm wird `FELDABREC 100 1` aufgerufen. Dieses ruft wiederum `FELDABREC 50 0` auf. Dieser Aufruf wird jedoch sofort wieder beendet, da `B` Null ist. Danach wird von `FELDABREC 100 1` weitere dreimal der Aufruf `FELDABREC 50 0` getätigt. Danach wird auch dieser Aufruf von `FELDABREC 100 1` beendet, da dieser die letzte Zeile erreicht hat. `FELDABREC 200 2` ruft nun noch weitere drei Mal `FELDABREC 100 1` auf, die wiederum je viermal den Aufruf `FELDABREC 50 0` tätigen.

$$R_1 = ()()()()$$
$$R_2 = (R_1)(R_1)(R_1)(R_1)$$

Aufgabe 13.35

	0	1	2	3	4	5	6	7	8	9
H	100	100	100	100	100	100	100	100	100	100
TIEF	3	3	3	3	3	3	3	3	3	3
H(BAUM 200/3 2)	-	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$
TIEF(BAUM 200/3 2)	-	2	2	2	2	2	2	2	2	2
H(BAUM 400/9 1)	-	-	$\frac{400}{9}$	$\frac{400}{9}$	$\frac{400}{9}$	$\frac{400}{9}$	$\frac{400}{9}$	-	$\frac{400}{9}$	$\frac{400}{9}$
TIEF(BAUM 400/9 1)	-	-	1	1	1	1	1	-	1	1
H(BAUM 800/27 0)	-	-	-	$\frac{800}{27}$	-	$\frac{800}{27}$	-	-	-	$\frac{800}{27}$
TIEF(BAUM 800/27 0)	-	-	-	0	-	0	-	-	-	0

	10	11	12	13	14	15	16	17	18	19
H	100	100	100	100	100	100	100	100	100	100
TIEF	3	3	3	3	3	3	3	3	3	3
H(BAUM 200/3)2	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	-	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$
TIEF(BAUM 200/3 2)	2	2	2	2	-	2	2	2	2	2
H(BAUM 400/9 1)	$\frac{400}{9}$	$\frac{400}{9}$	$\frac{400}{9}$	-	-	-	$\frac{400}{9}$	$\frac{400}{9}$	$\frac{400}{9}$	$\frac{400}{9}$
TIEF(BAUM 400/9 1)	1	1	1	-	-	-	1	1	1	1
H(BAUM 800/27 0)	-	$\frac{800}{27}$	-	-	-	-	-	$\frac{800}{27}$	-	$\frac{800}{27}$
TIEF(BAUM 800/27 0)	-	0	-	-	-	-	-	0	-	0

	20	21	22	23	24	25	26	27	28
H	100	100	100	100	100	100	100	100	100
TIEF	3	3	3	3	3	3	3	3	3
H(BAUM 200/3 2)	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	$\frac{200}{3}$	-
TIEF(BAUM 200/3 2)	2	2	2	2	2	2	2	2	2
H(BAUM 400/9 1)	$\frac{400}{9}$	-	$\frac{400}{9}$	$\frac{400}{9}$	$\frac{400}{9}$	$\frac{400}{9}$	$\frac{400}{9}$	-	-
TIEF(BAUM 400/9 1)	1	-	1	1	1	1	1	-	-
H(BAUM 800/27 0)	-	-	-	$\frac{800}{27}$	-	$\frac{800}{27}$	-	-	-
TIEF(BAUM 800/27 0)	-	-	-	0	-	0	-	-	-

Aufgabe 13.36

(a)

	0	1	2	3	4	5
H	120	120	120	120	120	120
TIEF	4	4	4	4	4	4
H(BAUM 80 3)	-	80	80	80	80	80
TIEF(BAUM 80 3)	-	3	3	3	3	3
H(BAUM 160/3 2)	-	-	$\frac{160}{3}$	$\frac{160}{3}$	$\frac{160}{3}$	$\frac{160}{3}$
TIEF(BAUM 160/3 2)	-	-	2	2	2	2
H(BAUM 320/9 1)	-	-	-	$\frac{320}{9}$	$\frac{320}{9}$	$\frac{320}{9}$
TIEF(BAUM 320/9 1)	-	-	-	1	1	1
H(BAUM 640/27 0)	-	-	-	-	$\frac{640}{27}$	-
TIEF(BAUM 640/27 0)	-	-	-	-	0	-

	6	7	8	9	10	11
H	120	120	120	120	120	120
TIEF	4	4	4	4	4	4
H(BAUM 80 3)	80	80	80	80	80	80
TIEF(BAUM 80 3)	3	3	3	3	3	3
H(BAUM 160/3 2)	$\frac{160}{3}$	$\frac{160}{3}$	$\frac{160}{3}$	$\frac{160}{3}$	$\frac{160}{3}$	$\frac{160}{3}$
TIEF(BAUM 160/3 2)	2	2	2	2	2	2
H(BAUM 320/9 1)	$\frac{320}{9}$	$\frac{320}{9}$	-	$\frac{320}{9}$	$\frac{320}{9}$	$\frac{320}{9}$
TIEF(BAUM 320/9 1)	1	1	-	1	1	1
H(BAUM 640/27 0)	$\frac{640}{27}$	-	-	-	$\frac{640}{27}$	-
TIEF(BAUM 640/27 0)	0	-	-	-	0	-

(b)

	0	1	2	3	4	5
H	243	243	243	243	243	243
TIEF	5	5	5	5	5	5
H(BAUM 162 4)	-	162	162	162	162	162
TIEF(BAUM 162 4)	-	4	4	4	4	4
H(BAUM 54 3)	-	-	54	54	54	54
TIEF(BAUM 54 3)	-	-	3	3	3	3
H(BAUM 36 2)	-	-	-	36	36	36
TIEF(BAUM 36 2)	-	-	-	2	2	2
H(BAUM 24 1)	-	-	-	-	24	24
TIEF(BAUM 24 1)	-	-	-	-	1	1
TIEF(BAUM 16 0)	-	-	-	-	-	16
TIEF(BAUM 16 0)	-	-	-	-	-	0

	6	7	8	9	10	11
H	243	243	243	243	243	243
TIEF	5	5	5	5	5	5
H(BAUM 162 4)	162	162	162	162	162	162
TIEF(BAUM 162 4)	4	4	4	4	4	4
H(BAUM 54 3)	54	54	54	54	54	54
TIEF(BAUM 54 3)	3	3	3	3	3	3
H(BAUM 36 2)	36	36	36	36	36	36
TIEF(BAUM 36 2)	2	2	2	2	2	2
H(BAUM 24 1)	24	24	24	-	24	24
TIEF(BAUM 24 1)	1	1	1	-	1	1
TIEF(BAUM 16 0)	-	16	-	-	-	16
TIEF(BAUM 16 0)	-	0	-	-	-	0

Aufgabe 13.38

```
to BAUM2 :H :TIEF
if :TIEF=0 [stop]
fd :H lt 45
BAUM2 :H/2 :TIEF-1
rt 45
BAUM2 :H*2/3 :TIEF-1
rt 45
BAUM2 :H/2 :TIEF-1
lt 45 bk :H
end
```

Kontrollaufgabe 1

```
to VIELSPIRREC :GR :ADD :ECK
fd :GR rt 360/:ECK
VIELSPIRREC :GR+:ADD :ADD :ECK
end
```

Kontrollaufgabe 2

```
to VIELSPIRREC2 :GR :ADD :ECK :MAX
if :GR>:MAX [stop]
fd :GR rt 360/:ECK
VIELSPIRREC2 :GR+:ADD :ADD :ECK :MAX
end
```

Kontrollaufgabe 3

```
to QUADINQUAD :GR :TIEF
if :TIEF=0 [stop]
fd :GR/2 rt 45
QUADINQUAD :GR*sqrt 0.5 :TIEF-1
lt 45 fd :GR/2 rt 90
repeat 3 [fd :GR rt 90]
end
```

Kontrollaufgabe 4

```
to DREIINDREI :GR :TIEF
if :TIEF=0 [stop]
repeat 3 [fd :GR  rt 120
          DREIINDREI :GR/2 :TIEF-1]
end
```

Kontrollaufgabe 5

```
to KREISREC2 :UM :TIEF :FAR
if :TIEF=0 [stop]
repeat 4 [KREISREC2 :UM/3 :TIEF-1 :FAR+1
          setpencolor :FAR
          repeat 90 [fd :UM/360 rt 1]]
end
```

Kontrollaufgabe 7

```
to FARBKREISE1 :UM
if :UM=0 [stop]
repeat 360 [fd :UM/360 rt 1]
FARBKREISE1 :UM-1
end

to FARBKREISE :UM :FAR
setpencolor :FAR
FARBKREISE1 :UM
if :FAR>16 [make "FAR 0]
FARBKREISE2 :UM :FAR+1
end
```

Kapitel 13

Aufgabe 14.1

```
to STRAHLENSATZ :alpha :SA1 :SA2 :SA3
if :alpha<90
  [rt  90 fd :SA3+30 bk 30
   lt 90 fd 1000 bk 1000 rt 90
   bk :SA3-:SA2
   lt 90 fd 1000 bk 1000 rt 90
   bk :SA2-:SA1
   lt 90 fd 1000 bk 1000 rt 90
   bk :SA1 lt :alpha fd 1000]
  [pr [alpha ist zu gross!]]
end
```

Aufgabe 14.2

```
to DREIAALPHA :a :alpha
if :alpha<90 [make "c :a/sin(:alpha)
              make "b :c*cos(:alpha)
              rt 90 fd :a lt 90 fd :b lt 180-:alpha fd :c]
             [pr [alpha ist zu gross]]
end
```

Mittels folgenden Formeln können die fehlenden Seiten berechnet werden:

$$c = \frac{a}{sin\alpha}$$
$$b = c \cdot cos(\alpha)$$

Da nun alle Seiten bekannt sind, kann das Dreieck gezeichnet werden.

Aufgabe 14.3

```
to DREIBALPHA :b :alpha
if :alpha<90 [make "c :b/cos(:alpha)
              make "a :c*sin(:alpha)
              rt 90 fd :a lt 90 fd :b lt 180-:alpha fd :c]
             [pr [alpha ist zu gross]]
end
```

Mittels folgenden Formeln können die fehlenden Seiten berechnet werden:

$$c = \frac{b}{cos\alpha}$$
$$a = c \cdot sin(\alpha)$$

Da nun alle Seiten bekannt sind, kann das Dreieck gezeichnet werden.

Aufgabe 14.5

```
to STRASSE :H :WIN
if :WIN<90 [make "STR :H/sin(:WIN)
                        make "A :STR*cos(:WIN)
                        rt 90 fd :A lt 90 fd :H lt 90+:WIN fd :STR]
                       [pr [WIN ist zu gross]]
end
```

Aufgabe 14.7

```
to ACOSINUS :X :APP
if :X=0 [pr [Fehler] stop]
if :X=1 [pr [Fehler] stop]
if :X<0 [pr [Fehler] stop]
if :X>1 [pr [Fehler] stop]
make "ALPHA :APP make "Y cos :ALPHA
while [:X<:Y ]
       [make "ALPHA :ALPHA+:APP make "Y cos :ALPHA]
make "ALPHA :ALPHA-:APP/2
pr :ALPHA fd 100 bk 100 lt :ALPHA fd 100
end
```

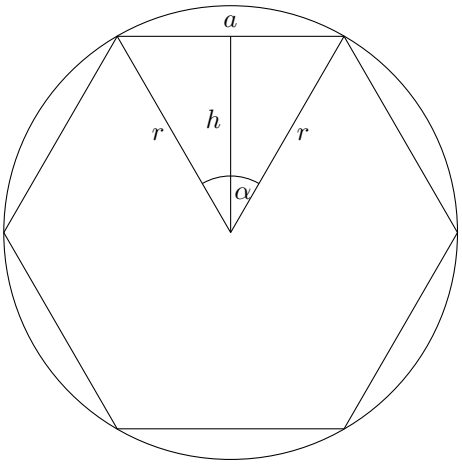
Der Aufbau ist derselbe wie der des Programmes ASINUS im Buch. Es ist jedoch zu beachten, dass der Cosinus für grösser werdenden Winkel immer kleiner wird. Deshalb muss auch die Bedingung von :X>:Y auf :X<:Y geändert werden.

Aufgabe 14.8

```
to DREIAB :A :B
make "BETA atan(:A/:B)
make "ALPHA atan(:B/:A)
make "C :A/sin(:ALPHA)
pr [beta=] pr :BETA
pr [alpha=] pr :ALPHA
pr [c=] pr :C
rt 90 fd :A lt 90 fd :B lt 180-:ALPHA fd :C
end
```

Die beiden fehlenden Winkel können mit dem Arcustangens berechnet werden. Die Seite c kann man dann mit dem Sinus von α berechnen.

Aufgabe 14.10 Am Beispiel eines Sechsecks wird die Herleitung der Formel für die Fläche eines regelmässigen n-Eckes erklärt.



Die Fläche eines n-Eckes ist n-mal die Fläche des Dreiecks, dessen Seiten aus einer Seite des n-Eckes und aus dem Radius bestehen (s. Skizze).

$$A_{n-Eck} = n \cdot A_{\triangle} = n \cdot \frac{a \cdot h}{2}$$

Nun muss die Höhe h sowie die Seite a ermittelt werden:

$$cos\left(\frac{\alpha}{2}\right) = \frac{h}{r} \qquad sin\left(\frac{\alpha}{2}\right) = \frac{\frac{a}{2}}{r}$$
$$h = r \cdot cos\left(\frac{\alpha}{2}\right) \qquad \frac{a}{2} = r \cdot sin\left(\frac{\alpha}{2}\right)$$
$$a = 2r \cdot sin\left(\frac{\alpha}{2}\right)$$

Eingesetzt in die Formel für die Fläche sieht das Ganze so aus:

$$A_{n-Eck} = n \cdot \frac{2r \cdot sin(\frac{\alpha}{2}) \cdot r \cdot cos(\frac{\alpha}{2})}{2}$$
$$= n \cdot \left[r \cdot sin\left(\frac{\alpha}{2}\right) \cdot r \cdot cos\left(\frac{\alpha}{2}\right)\right]$$

Der Winkel α beträgt $\frac{360}{n}$, da das Dreieck genau n-mal im n-Eck Platz hat. Also ergibt sich:

$$A_{n-Eck} = n \cdot \left[r \cdot sin\left(\frac{180}{n}\right) \cdot r \cdot cos\left(\frac{180}{n}\right)\right]$$

Nun da die Formel für die Fläche eines regelmässigen n-Ecks bekannt ist, kann das Programm geschrieben werden

```
to FLVIELECK :R :N
if :R<0 [pr [Fehler] stop]
if :N<3 [pr [Fehler] stop]
make "A :N*(:R*:R*sin(180/:N)*cos(180/:N))
pr [Flaeche=] pr :A
end
```

Aufgabe 14.11 Um die fehlenden Seiten zu berechnen werden folgende Beziehungen genutzt:

$$cos\left(\frac{\alpha}{2}\right) = \frac{h}{b} \qquad sin\left(\frac{\alpha}{2}\right) = \frac{\frac{a}{2}}{b}$$
$$b = \frac{h}{cos(\frac{a}{2})} \qquad a = 2b \cdot sin\left(\frac{\alpha}{2}\right)$$

```
to FLDREI :H :ALPHA
if :ALPHA<0 [pr [Fehler]stop]
make "B :H/cos(:ALPHA/2)
make "A 2*:B*sin(:ALPHA/2)
make "BETA (180-:ALPHA)/2
make "FL :A*:H/2
rt 90 fd :A lt 180-:BETA fd :B lt 180-:ALPHA fd :B
pr [Flaeche=] pr :FL
end
```

Kontrollaufgabe 1

```
to STRAHLENSATZ :ALPHA :SA1 :SA2 :SA3
if :SA1>:SA2 [pr [SA1 zu gross] stop]
if :SA2>:SA3 [pr [SA2 zu gross] stop]
make "SB3 :SA3/cos :ALPHA
make "A1B1 :SA1*tan :ALPHA
make "A2B2 :SA2*tan :ALPHA
make "A3B3 :SA3*tan :ALPHA
rt 90-:ALPHA fd :SB3+10 bk :SB3+10 rt :ALPHA
fd :SA3+10 bk 10
lt 90 fd :A3B3 bk :A3B3 rt 90
bk :SA3-:SA2
lt 90 fd :A2B2 bk :A2B2 rt 90
bk :SA2-:SA1
lt 90 fd :A1B1 bk :A1B1 rt 90
bk :SA1 lt 90
end
```

Die fehlenden Seiten können mit dem Cosinus und dem Tangens berechnet werden. Danach wird zuerst die Strecke $\overline{SB_3}$ gezeichnet und dann die Strecke SA_3. Nun können die fehlenden Strecken $\overline{A_3B_3}$, $\overline{A_2B_2}$ und $\overline{A_1B_1}$ gezeichnet werden.

Kontrollaufgabe 2

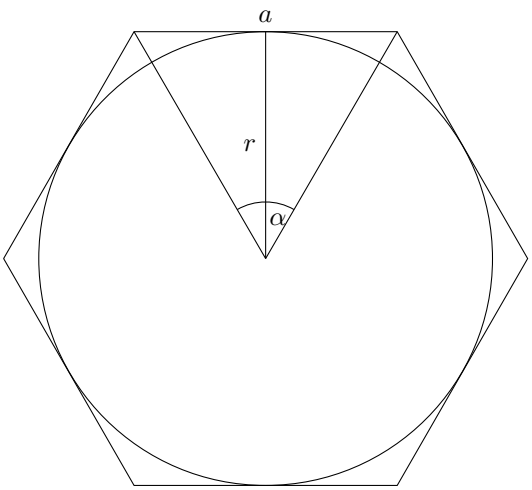
```
to DREIALLES :BETA :C
if :BETA>90 [pr [BETA zu gross]stop]
if :BETA=90 [pr [BETA zu gross]stop]
make "ALPHA 90-:BETA
make "B :C*sin :BETA
make "A :C*sin :ALPHA
rt 90 fd :A lt 90 fd :B lt 180-:ALPHA fd :C rt 90+:BETA
end
```

Kontrollaufgabe 3 Das Programm kann so aufgebaut werden wie das Programm ASINUS. Beim Tangens kann das Argument jedoch auch grösser als 1 sein, was beim Sinus und Cosinus nicht möglich ist.

```
to ARCTANGENS :X :ADD
if :X<0 [pr [X muss positiv sein]stop]
make "ALPHA :ADD make "Y tan :ALPHA
while [:X>:Y]
         [make "ALPHA :ALPHA+:ADD make "Y tan :ALPHA]
```

```
make "ALPHA :ALPHA-:ADD/2
pr :ALPHA fd 100 bk 100 lt :ALPHA fd 100
end
```

Kontrollaufgabe 4



Um die Fläche zu berechnen wird n-mal die Fläche des in der Skizze eingezeichneten Dreiecks berechnet, da dieses Dreieck n-mal in dem n-Eck Platz findet. Die Fläche des n-Ecks beträgt also:

$$A_{n-Eck} = n \cdot A_{\triangle} = n \cdot \frac{a \cdot r}{2}$$

Der Radius ist gegeben und die Seite a kann mit Hilfe des Tangens berechnet werden:

$$tan\left(\frac{\alpha}{2}\right) = \frac{\frac{a}{2}}{r}$$
$$a = 2r \cdot tan\left(\frac{\alpha}{2}\right)$$

Diese Erkenntnis kann sogleich in die Flächenformel eingesetzt werden.

$$A_{n-Eck} = n \cdot r^2 \cdot tan\left(\frac{\alpha}{2}\right)$$

Der Winkel α ist bekannt. Er beträgt $\frac{360}{n}$. Also lautet die Formel:

$$A_{n-Eck} = n \cdot r^2 \cdot tan\left(\frac{180}{n}\right)$$

Nun kann ein Programm geschrieben werden, das diese Fläche mit Hilfe der gefundenen Formel berechnet.

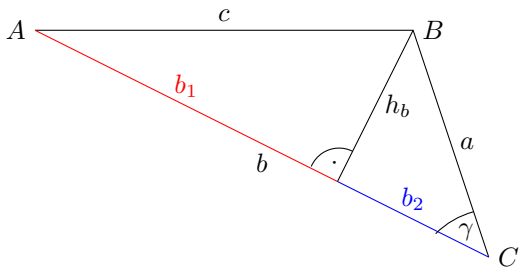
```
to FLVIELECK2 :R :N
if :N<3 [pr [N zu klein]stop]
make "FL :N*:R*:R*tan(180/:N)
pr [Flaeche] pr :FL
end
```

Kontrollaufgabe 5

```
to PIBERECHNEN :N
if :N<3 [pr [N zu klein]stop]
make "PI :N*tan(180/:N)
pr [Pi] pr :PI
end
```

Kontrollaufgabe 6 In dieser Aufgabe müssen drei Fälle unterschieden werden. Entweder ist γ kleiner oder grösser als 90 Grad oder genau 90 Grad.

Um c zu berechnen für γ kleiner als 90 Grad wird so vorgegangen:



Um die Länge des Tunnels c zu ermitteln, wird zuerst die Höhe h_b ermittelt.

$$sin\gamma = \frac{h_b}{a}$$
$$h_b = a \cdot sin\gamma$$

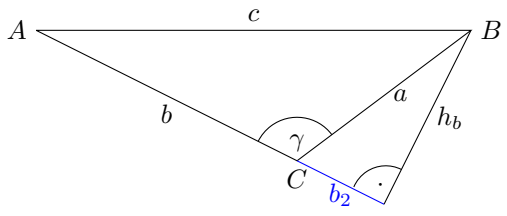
Damit kann nun b_2 ermittelt werden (s. Skizze).

$$cos\gamma = \frac{b_2}{a}$$
$$b_2 = a \cdot cos\gamma$$

Die Seite c kann nun mit dem Satz des Pythagoras ermittelt werden, wobei $b_1 = b - b_2$ gilt:

$$c = \sqrt{b_1^2 + h_b^2}$$
$$c = \sqrt{(b - b_2)^2 + h_b^2}$$

Falls γ kleiner als 90 Grad ist, muss so vorgegangen werden:



Dioe Höhe h_b beträgt in diesem Fall:

$$sin(180 - \gamma) = \frac{h_b}{a}$$
$$h_b = a \cdot sin(180 - \gamma)$$

Auch die Strecke b_2 muss anders berechnet werden:

$$cos(180 - \gamma) = \frac{b_2}{a}$$
$$b_2 = a \cdot cos(180 - \gamma)$$

Die Seite c kann nun folgendermassen bestimmt werden:

$$c = \sqrt{(b + b_2)^2 + h_b^2}$$

Wenn der Winkel genau 90 Grad beträgt, ist c:

$$c = \sqrt{a^2 + b^2}$$

Das Programm hat nun die Aufgabe c mit diesen Formel zu ermitteln.

```
to TUNNEL :A :B :GAMMA
if :GAMMA<0 [pr [Gamma zu klein]stop]
if :GAMMA>180 [pr [Gamma zu gross]stop]
if :GAMMA<90 [make "HB :A*sin :GAMMA
              make "B2 :A*cos :GAMMA
              make "C sqrt((:B-:B2)*(:B-:B2)+:HB*:HB)
              pr [Tunnellaenge] pr :C]
if :GAMMA=90 [make "C sqrt(:A*:A+:B*:B)
              pr [Tunnellaenge] pr :C]
if :GAMMA>90 [make "GAMMA1 180-:GAMMA
              make "HB :A*sin :GAMMA1
              make "B2 :A*cos :GAMMA1
              make "C sqrt((:B+:B2)*(:B+:B2)+:HB*:HB)
             pr [Tunnellaenge] pr :C]
end
```

Kontrollaufgabe 7

(a) Zuerst wird `X` auf den Startwert von 0 gesetzt und die Funktion von 0 berechnet und in `FMAX` gespeichert. Nun werden mittels einer `while`-Schleife alle Funktionswerte für die Argumente 0 bis 90 berechnet und mit `FMAX` verglichen. Ist ein Funktionswert grösser als `FMAX` so wird er als neues `FMAX` gespeichert. Zusätzlich wird die Variable `XMAX` gespeichert, in der der Wert von dem `X` gespeichert wird, das für die Berechnung von `FMAX` verwendet wurde.

```
to FUNALPHA :APP
if :APP=0 [pr [APP zu klein]stop]
if :APP<0 [pr [APP zu klein]stop]
make "X :APP
make "FMAX sin :X*cos :X
make "XMAX :X
while [:X<90] [make "F sin :X*cos :X
               make "X :X+:APP
               if :F>:FMAX [make "FMAX :F make "XMAX :X]]
pr [Maximum bei] pr :XMAX
pr [FMAX=] pr :FMAX
end
```

(b) Das Programm hat das gleiche Konzept wie das Programm in a) aber es wird eine andere Funktion ausgeführt.

```
to FUNX :APP
if :APP=0 [pr [APP zu klein]stop]
if :APP<0 [pr [APP zu klein]stop]
make "X (-10)
make "FMAX (20+:X)*(20-:X)
while [:X<10] [make "F (20+:X)*(20-:X)
               make "X :X+:APP
               if :F>:FMAX [make "FMAX :F make "XMAX :X]]
pr [Maximum bei] pr :XMAX
pr [FMAX] pr :FMAX
end
```

(c) Das Programm funktioniert nach dem gleichen Prinzip wie die Programme aus a) und b), nur die Funktion wurde geändert.

```
to FUNX2 :A :B :C :D :APP
if :APP=0 [pr [APP zu klein]stop]
if :APP<0 [pr [APP zu klein]stop]
make "X 0
make "FMAX :A*:X*:X*:X+:B*:X*:X+:C*:X+:D
make "XMAX :X
while [:X<100] [make "F :A*:X*:X*:X+:B*:X*:X+:C*:X+:D
                make "X :X+:APP
                if :F>:FMAX [make "FMAX :F make "XMAX :X]]
pr [Maximum bei] pr :XMAX
pr [FMAX] pr :FMAX
end
```

Kapitel 14

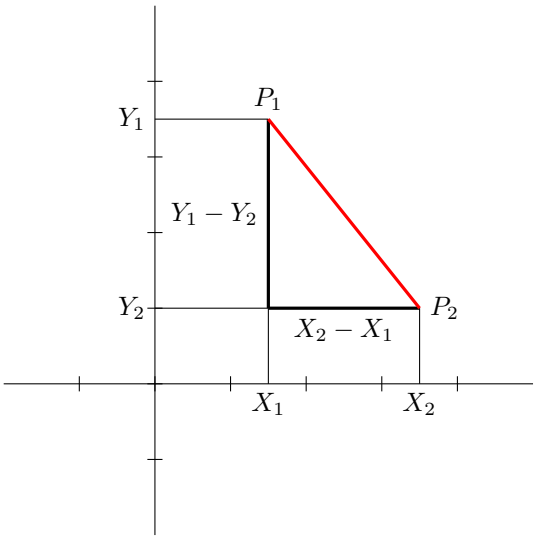
Aufgabe 15.1 Das Programm KOOR2 hat im Unterschied zu KOOR zwei Parameter mehr, damit zwei Parameter die Anzahl und Länge der Einheiten auf der x-Achse bestimmen können und die anderen zwei die Anzahl und die Länge der Einheiten der y-Achse

```
to KOOR2 :AN1 :EINHEIT1 :AN2 :EINHEIT2
ACHSE :AN2 :EINHEIT2
rt 90
ACHSE :AN1 :EINHEIT1
lt 90
end
```

Aufgabe 15.2 Nach dem Zeichnen des zweidimensionalen Koordinatensystems wird eine weitere Achse im Winkel von 45 Grad eingezeichnet. Nun kann der Punkt und der Quader gezeichnet werden.

```
to PUNKT3D :EINHEIT :X :Y :Z
KOOR 240/:EINHEIT :EINHEIT
rt 45
ACHSE 240/:EINHEIT :EINHEIT*2/3
lt 45
fd :Y*:EINHEIT rt 45
setpencolor red
fd :Z*:EINHEIT*2/3 rt 45
repeat 2 [fd :X*:EINHEIT rt 90 fd :Y*:EINHEIT rt 90]
fd :X*:EINHEIT
KREISE 1/36
rt 135
repeat 2 [fd :Z*:EINHEIT*2/3 lt 45 fd :Y*:EINHEIT lt 135]
fd :Z*:EINHEIT*2/3 rt 45 fd :X*:EINHEIT
end
```

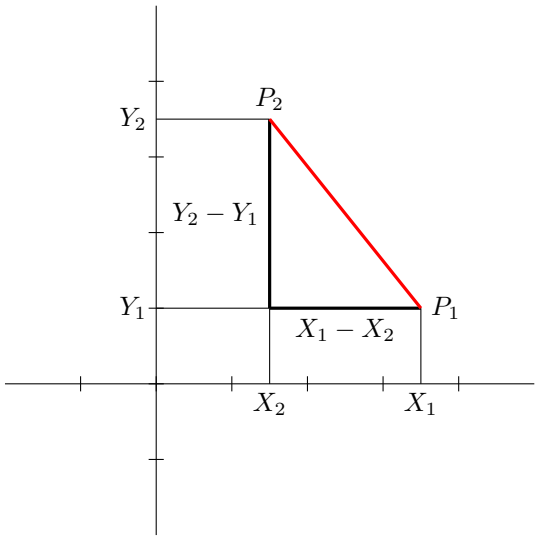
Aufgabe 15.3 Wenn $Y_2 \leq Y_1$ gilt, dann muss nach rechts gedreht werden, da die zu zeichnende Strecke von links oben nach rechts unten verlaufen soll. Würde nach links gedreht, dann würden nicht beide Punkte durch die Strecke verbunden werden, da die Strecke vom Punkt (X_1, Y_1) nach oben rechts weggehen würde statt nach unten rechts, wo sich der zweite Punkt befindet.



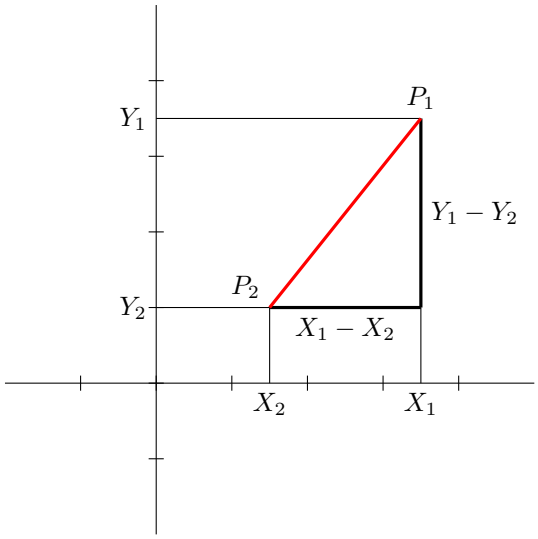
Aufgabe 15.4 Das Programm STRECKE muss so erweitert werden, dass auch für $X_1 > X_2$ der korrekte Winkel verwendet wird. Im ursprünglichen Programm wurde der Winkel immer als Arkustangens des Betrages der Differenz $X_2 - X_1$ berechnet. Dieser Winkel liegt immer zwischen 0 und 90 Grad. Ist jedoch $X_1 > X_2$, muss der Winkel $180 - arctan(|X_2 - X_1|)$ betragen. Diese Option muss also noch mit einem if-Befehl ermöglicht werden

```
to STRECKE2 :EINHEIT :X1 :Y1 :X2 :Y2
setpc [0 0 0]
PUNKT :EINHEIT :X2 :Y2
wait 50 pu bk :X2*:EINHEIT lt 90 bk :Y2*:EINHEIT pd
PUNKT :EINHEIT :X1 :Y1
make "DIST sqrt ((:X2-:X1)*(:X2-:X1)+(:Y2-:Y1)*(:Y2-:Y1))
pr :DIST
if :Y2>:Y1 [make "ABSDY :Y2-:Y1]
           [make "ABSDY :Y1-:Y2]
if :X1=:X2 [make "ALPHA 90]
if :X1<:X2 [make "ALPHA atan (:ABSDY/abs(:X2-:X1))]
if :X2<:X1 [make "ALPHA 180- atan (:ABSDY/abs(:X2-:X1))]
setpc [255 0 0]
if :Y2>:Y1 [lt :ALPHA] [rt :ALPHA]
fd :DIST*:EINHEIT
end
```

Das Bild für $X_1 > X_2$ und $Y_2 > Y_1$ sieht so aus:



Für $X_1 > X_2$ und $Y_1 > Y_2$ sieht die Abbildung so aus:



Aufgabe 15.5 Um die Schildkröte an ihre Startposition zu bringen, muss sie zurück zum Punkt P_1 gebracht werden. Dort wird Sie um den Winkel ALPHA zurückgedreht. Nun kann sie im Wandermodus um die Strecke X_1 versetzt werden, und zwar so, dass sie auf der y-Achse zu liegen kommt. Danach muss sie nur noch gedreht und um die Strecke Y_1 an den Startpunkt zurückgesetzt werden. Wichtig ist dabei, dass mit einer Verzweigung kontrolliert wird, dass auf die richtige Seite gedreht wird. Am besten geht dies, indem man wie beim Zeichnen die beiden Werte Y_1 und Y_2 vergleicht. Ist Y_2 grösser, muss nach rechts gedreht werden und sonst nach links.

```
to STRECKE3 :EINHEIT :X1 :X2 :Y1 :Y2
setpc [0 0 0]
PUNKT :EINHEIT :X2 :Y2
wait 50 pu bk :X2*:EINHEIT lt 90 bk :Y2*:EINHEIT pd
PUNKT :EINHEIT :X1 :Y1
make "DIST sqrt ((:X2-:X1)*(:X2-:X1)+(:Y2-:Y1)*(:Y2-:Y1))
pr :DIST
```

```
if :Y2>:Y1 [make "ABSDY :Y2-:Y1]
           [make "ABSDY :Y1-:Y2]
if :X1=:X2 [make "ALPHA 90]
if :X1<:X2 [make "ALPHA atan (:ABSDY/abs(:X2-:X1))]
if :X2<:X1 [make "ALPHA 180- atan (:ABSDY/abs(:X2-:X1))]
setpc [255 0 0]
if :Y2>:Y1 [lt :ALPHA] [rt :ALPHA]
fd :DIST*:EINHEIT
pu bk :DIST*:EINHEIT
if :Y2>:Y1 [rt :ALPHA] [lt :ALPHA]
bk :X1*:EINHEIT lt 90 bk :Y1*:EINHEIT pd
setpc [0 0 0]
end
```

Aufgabe 15.6 Die Berechnung des Schnittpunkt erfolgt, indem die Koordinatenform der beiden Geraden einander gleichgesetzt werden und die entstandene Gleichung nach x aufgelöst wird.

$$g_1(x) = x \cdot m_1 + n_1 \qquad g_2(x) = x \cdot m_2 + n_2$$

$$\begin{aligned} x \cdot m_1 + n_1 &= x \cdot m_2 + n_2 \\ x \cdot m_1 - x \cdot m_2 &= n_2 - n_1 \\ x &= \frac{n_2 - n_1}{m_1 - m_2} \end{aligned}$$

Die Parameter m_1, m_2, n_1 und n_2 können alle durch die gegebenen Koordinaten ausgedrückt werden. Die Steigungen sind definiert als:

$$m_1 = \frac{y_2 - y_1}{x_2 - x_1} \qquad m_2 = \frac{y_4 - y_3}{x_4 - x_3}$$

Die Achsenabschnitte der y-Achse können wie folgt ermittelt werden:

$$y_1 = x_1 \cdot m_1 + n_1 \qquad y_3 = x_3 \cdot m_2 + n_2$$
$$n_1 = y_1 - x_1 \cdot m_1 \qquad n_2 = y_3 - x_3 \cdot m_3$$

Das Programm zum Zeichnen der Geraden ist eine modifizierte Version des Programmes STRECKE.

```
to GERADEN :EINHEIT :X1 :Y1 :X2 :Y2 :X3 :Y3 :X4 :Y4
make "A1 :X1 make "B1 :Y1
make "A2 :X2 make "B2 :Y2
repeat 2 [setpc [0 0 0]
    PUNKT :EINHEIT :A2 :B2
    wait 50 pu bk :A2*:EINHEIT lt 90 bk :B2*:EINHEIT pd
    PUNKT :EINHEIT :A1 :B1
    if :B2>:B1 [make "ABSDY :B2-:B1]
               [make "ABSDY :B1-:B2]
    if :A1=:A2 [make "ALPHA 90]
    if :A1<:A2 [make "ALPHA atan (:ABSDY/abs(:A2-:A1))]
    if :A2<:A1 [make "ALPHA 180- atan (:ABSDY/abs(:A2-:A1))]
    setpc [0 0 128]
```

```
    if :B2>:B1 [lt :ALPHA] [rt :ALPHA]
    fd 500*:EINHEIT bk 1000*:EINHEIT fd 500*:EINHEIT
    pu
    if :B2>:B1 [rt :ALPHA] [lt :ALPHA]
    bk :A1*:EINHEIT lt 90 bk :B1*:EINHEIT pd
    make "A1 :X3 make "B1 :Y3
    make "A2 :X4 make "B2 :Y4]
make "M1 (:Y2-:Y1)/(:X2-:X1)
make "M2  (:Y4-:Y3)/(:X4-:X3)
make "N1 :Y1-:X1*:M1
make "N2 :Y3-:X3*:M2
if :M1=:M2 [if :N1=:N2[pr [Die Geraden liegen aufeinander]stop]
                      [pr [Die Geraden sind parallel]stop]]
make "X (:N2-:N1)/(:M1-:M2)
make "Y :X*:M1+:N1
pr [X=] pr :X
pr [Y=] pr :Y
end
```

Die `make`-Befehle dienen dazu, dass trozt der `repeat`-Schleife verschiedene Variablen verwendet werden können.

Aufgabe 15.7 Um den Schnittpunkt zu berechnen werden die Gleichungen der Geraden gleichgesetzt:

$$\begin{aligned} A \cdot x + B &= C \cdot x + D \\ A \cdot x - C \cdot x &= D - B \\ x(A - C) &= D - B \\ x &= \frac{D - B}{A - C} \end{aligned}$$

Die y-Koordinate des Schnittpunktes erhält man, wenn x in eine der Geradengleichungen eingesetzt wird.

Für die grafische Darstellung wird mit Hilfe der Gleichung $m = arctan(\alpha)$ der Winkel berechnet, der sich zwischen der x-Achse und der Geraden befindet.

```
to GERADEN2 :EINHEIT :A :B :C :D
KOOR 10 :EINHEIT
make "ALPHA1 atan :A
make "ALPHA2 atan :C
setpc [0 0 128]
fd :B*:EINHEIT rt 90-:ALPHA1
fd 500*:EINHEIT bk 1000*:EINHEIT fd 500*:EINHEIT
lt 90-:ALPHA1 bk :B*:EINHEIT
fd :D*:EINHEIT rt 90-:ALPHA2
fd 500*:EINHEIT bk 1000*:EINHEIT fd 500*:EINHEIT
lt 90-:ALPHA2 bk :D*:EINHEIT
if :A=:C [if :B=:D[pr [Die Geraden liegen aufeinander]stop]
                      [pr [Die Geraden sind parallel]stop]]
make "X (:D-:B)/(:A-:C)
make "Y :A*:X+:B
pr [X=] pr :X
pr [Y=] pr :Y
end
```

Aufgabe 15.8 Zu Beginn des Programmes wird der Winkel $\angle P_3P_1P_2$ berechnet. Falls dieser Null beträgt, liegen alle drei Punkte auf der gleichen Gerade. Würden nämlich nicht auf der gleichen Gerade liegen, dann würden sie ein Dreieck aufspannen und dann kann $\angle P_3P_1P_2$ nicht Null sein.

Falls alle Punkte auf einer Gerade liegen, werden sie zuerst eingezeichnet und danach wird eine Gerade durch alle drei Punkte gezogen.

Falls die drei Punkte nicht auf einer Gerade liegen, dann wird das Dreieck mit Hilfe des Programmes STRECKEN3 gezeichnet.

```
to KOOR3PUNKTE :EINHEIT :X1 :Y1 :X2 :Y2 :X3 :Y3
make "ALPHA atan (abs((:Y2-:Y1)/(:X2-:X1)-(:Y3-:Y1)/(:X3-:X1)))
if :ALPHA=0 [make "ALPHA1 atan ((:Y2-:Y1)/(:X2-:X1))
             STRECKE3 :EINHEIT :X1 :Y1 :X2 :Y2
             STRECKE3 :EINHEIT :X2 :Y2 :X3 :Y3
             pu fd :Y1*:EINHEIT rt 90 fd :X1*:EINHEIT
             lt 90-:ALPHA1 pd setpc [0 0 128]
             fd 500*:EINHEIT bk 1000*:EINHEIT
             fd 500*:EINHEIT setpc [0 0 0]]
            [STRECKE3 :EINHEIT :X1 :Y1 :X2 :Y2
             STRECKE3 :EINHEIT :X2 :Y2 :X3 :Y3
             STRECKE3 :EINHEIT :X3 :Y3 :X1 :Y1]
end
```

Aufgabe 15.9 Die Gleichung für die Gerade ist in der Aufgabe bereits gegeben. Die Gleichung des Kreises lautet:

$$(x - XK)^2 + (y - YK)^2 = R^2$$

In diese Gleichung wird y durch $Ax + B$ aus der Geradengleichung ersetzt. Dann kann nach x aufgelöst werden.

$$\begin{aligned}
(x - XK)^2 + (Ax + B - YK)^2 &= R^2 \\
x^2 - 2x \cdot XK + XK^2 + A^2x^2 + 2Ax(B - YK) + (B - YK)^2 - R^2 &= 0 \\
A^2x^2 + x^2 + 2Ax(B - YK) - 2x \cdot XK + XK^2 + (B - YK)^2 - R^2 &= 0 \\
x^2(A^2 + 1) + x(2A[B - YK] - 2XK) + XK^2 + (B - YK)^2 - R^2 &= 0
\end{aligned}$$

Nun könnte die Lösungsformel für quadratische Gleichungen angewendet werden. Dies würde aber zu einem komplizierten Ausdruck führen und deshalb wird hier substiuiert mit $L_1 = A^2 + 1$, $L_2 = 2A(B - YK) - 2XK$ und $L_3 = XK^2 + (B - YK)^2 - R^2$. Danach wird in die Lösungsformel eingesetzt.

$$L_1x^2 + L_2x + L_3 = 0$$

$$x = \frac{-L_2 \pm \sqrt{(L_2)^2 - 4L_1L_3}}{2L_1}$$

Das Programm zeichnet zuerst die Lösung. Dabei wird das Programm KREISMITT aus dem Buch auf Seite 198 verwendet. Nach dem Zeichnen werden die drei Variabeln L_1, L_2 und L_3 erstellt, die weiter oben zur Substitution verwendet wurden. Danach wird mit der die Diskriminante berechnet und dann, je nach Wert der Diskriminante, werden Schnittpunkte berechnet.

```
to KREISGERADE :XK :YK :R :A :B
KOOR 10 30
make "ALPHA atan :A
fd :B*30 rt :ALPHA
```

```
fd 500 bk 1000 fd 500
lt :ALPHA bk :B*30
pu fd :YK*30 rt 90 fd :XK*30 pd
KREISMITT :R*30
make "L1 (:A*:A+1)
make "L2 (2*:A*(:B-:YK)-2*:XK)
make "L3 (:XK*:XK-:R*:R+(:B-:YK)*(:B-:YK))
make "D :L2*:L2-4*:L1*:L3
if :D<0 [pr [keine Schnittpunkte]stop]
if :D=0 [make "X (-:L2)/(2*:L1)
         make "Y :A*:X+:B
         pr [X=] pr :X pr [Y=] pr :Y stop]
if :D>0 [make "X1 ((-:L2)+sqrt :D)/(2*:L1)
         make "X2 ((-:L2)-sqrt :D)/(2*:L1)
         make "Y1 :A*:X1+:B
         make "Y2 :A*:X2+:B
         pr [X1=] pr :X1 pr [Y1=] pr :Y1
         pr [X2=] pr :X2 pr [Y2=] pr :Y2]
end
```

Aufgabe 15.11

```
to EINHEITSKREIS :X
if :X>1 [pr [X zu gross]stop]
if :X<0 [pr [X zu klein]stop]
fd 160 bk 320 fd 160
rt 90
fd 160 bk 320 fd 160
KREISMITT 150
make "ALPHA acos (:X)
make "Y sin :ALPHA
fd :X*150
setpc [0 0 128]
lt 90 fd :Y*150
setpc [0 150 150]
lt 90 fd :X*150 bk :X*150
setpc [255 0 0]
home
setpc [0 0 0]
end
```

In SUPERLOGO muss das Programm `ACOSINUS` aus verwendet werden, da der Befehl `acos` nicht existiert.

Aufgabe 15.12

(a) Die fehlenden Grössen α und X werden mit $\alpha = arcsin(Y)$ und $X = cos(\alpha)$ berechnet.

```
to EINHEITSKREIS2 :Y
if :Y>1 [pr [Y zu gross]stop]
if :Y<0 [pr [Y zu klein]stop]
fd 160 bk 320 fd 160
rt 90
fd 160 bk 320 fd 160
```

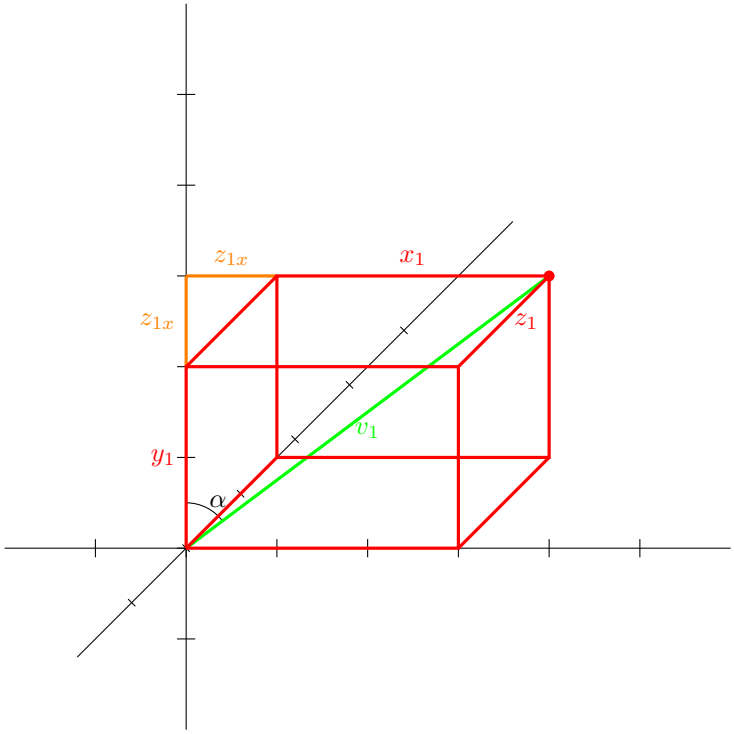
```
KREISMITT 150
make "ALPHA asin :Y
make "X cos :ALPHA
fd :X*150
setpc [0 0 128]
lt 90 fd :Y*150
setpc [0 150 150]
lt 90 fd :X*150 bk :X*150
setpc [255 0 0]
home
setpc [0 0 0]
end
```

In SUPERLOGO muss das Programm `ASINUS` verwendet werden, da der Befehl `asin` nicht existiert.

(b) Um X zu erhalten, nimmt man $cos(\alpha)$. Für Y nimmt man $\sin(\alpha)$.

```
to EINHEITSKREIS3 :ALPHA
if :ALPHA>90 [pr [ALPHA zu gross]stop]
if :ALPHA<0 [pr [ALPHA zu klein]stop]
fd 160 bk 320 fd 160
rt 90
fd 160 bk 320 fd 160
KREISMITT 150
make "Y sin :ALPHA
make "X cos :ALPHA
fd :X*150
setpc [0 0 128]
lt 90 fd :Y*150
setpc [0 150 150]
lt 90 fd :X*150 bk :X*150
setpc [255 0 0]
home
setpc [0 0 0]
```

Aufgabe 15.13



Die roten Quader (s. Skizze oben) kann mit dem Programm `PUNKT3D` aus **??** gezeichnet werden. Um den Winkel α und die Läng des Vektors v_1 zu ermitteln, bedienen wir uns der Trigonometrie. α wird wie folgt ermittelt:

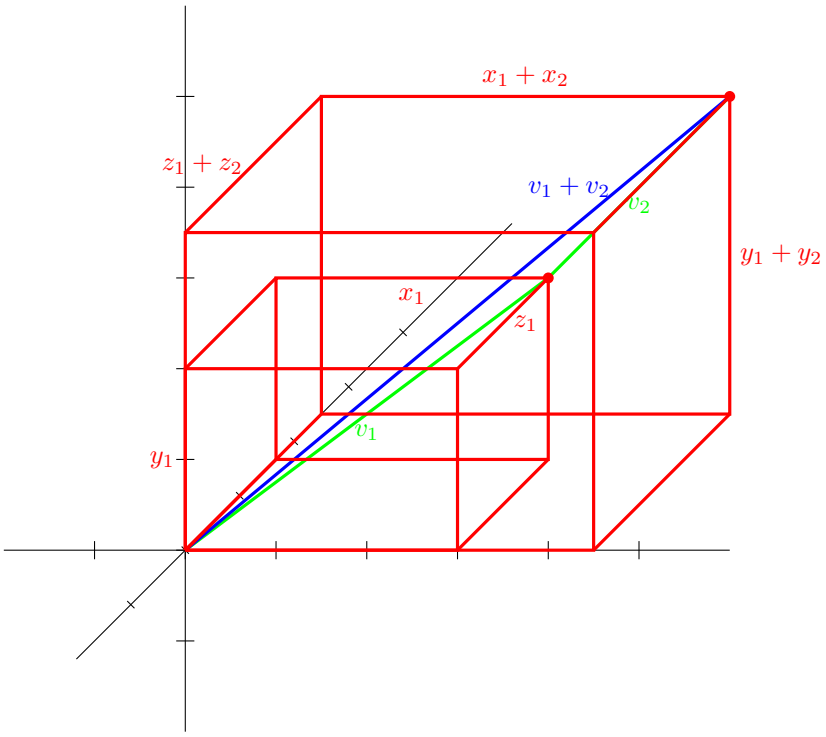
$$tan(\alpha) = \frac{x_1 + z_{1x}}{y_1 + z_{1x}}$$
$$\alpha = arctan\left(\frac{x_1 + z_{1x}}{y_1 + z_{1x}}\right)$$

v_1 wird durch den Satz des Pythagoras berechnet:

$$v_1 = \sqrt{(x_1 + z_{1x})^2 + (y_1 + z_{1x})^2}$$

Die Variable z_{1x} hat den Wert $\frac{1}{2}z_1\sqrt{2}$, da z_1 die Hypothenuse eines Dreiecks mit zwei gleich langen Katheten der Länge z_{1x} bildet.

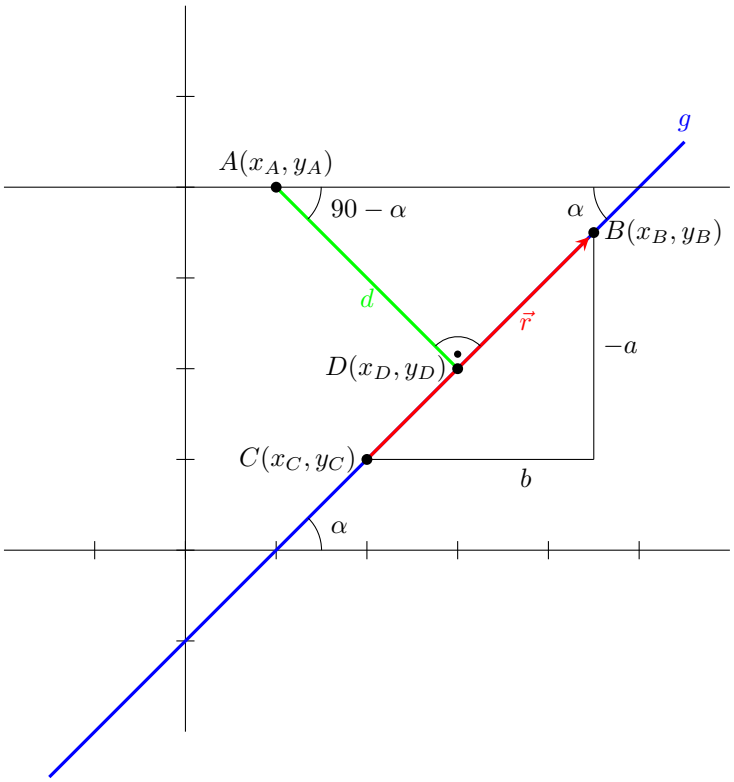
Für den zweiten Vektor geht man analog vor, einfach mit den Werten x_2, y_2 z_{2x}, β und v_2 Beim Zeichnen des zweiten Quaders muss jedoch darauf geachtet werden, dass dieser die Seitenlängen $x_1 + x_2$, $y_1 + y_2$ und $z_1 + z_2$ besitzt, damit die beiden Vektoren grafisch addiert werden (s. Skizze unten).

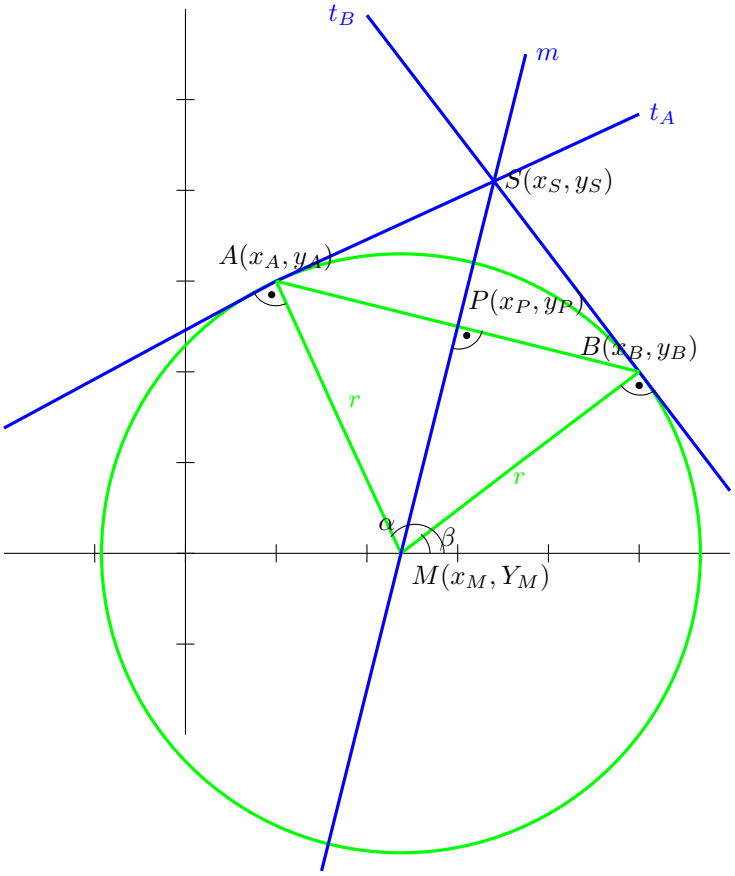


Nun werden alle diese Überlegungen in ein Programm gebracht:

```
to VEKTOREN3D :X1 :Y1 :Z1 :X2 :Y2 :Z2
PUNKT3D 30 :X1 :Y1 :Z1
setpc [0 0 0]
rt 90 bk :Y1*30
PUNKT3D 30 :X2+:X1 :Y2+:Y1 :Z2+:Z1
setpc [0 0 0]
rt 90 bk (:Y2+:Y1)*30
setpc [0 255 0]
make "Z1X 2/3*:Z1*sin 45
make "Z2X 2/3*:Z2*sin 45
make "ALPHA atan (:X1+:Z1X)/(:Y1+:Z1X)
make "BETA atan (:X2+:Z2X)/(:Y2+:Z2X)
make "V1 sqrt ((:X1+:Z1X )*(:X1+:Z1X )+(:Y1+:Z1X )*(:Y1+:Z1X ))
make "V2 sqrt ((:X2+:Z2X )*(:X2+:Z2X )+(:Y2+:Z2X )*(:Y2+:Z2X ))
rt :ALPHA fd :V1*30 lt :ALPHA
rt :BETA fd :V2*30
setpc [0 0 128]
home
end
```

Kontrollaufgabe 1


g
A(x_A, y_A)
90 − α
α
B(x_B, y_B)
d
r⃗
−a
D(x_D, y_D)
C(x_C, y_C)
b
α



Kontrollaufgabe 2

Um den Mittelpunkt des Kreises zu ermitteln, muss zuerst der Punkt P in der Mitte der Strecke zwischen den Punkten A und B ermittelt werden. Dieser Punkt trägt die Koordinaten:

$$x_p = \frac{x_A - x_B}{2} + x_B \qquad y_P = \frac{y_A - y_B}{2} + y_B$$

Nun kann die Mittelsenkrechte m der Strecke $\overline{AB}$ konstruiert werden. Als Stützpunkt wird P gewählt. Als Ortsvektor dient der negative Kehrwert des Vektors $\vec{AB}$, da diese senkrecht auf m steht.

$$m : \vec{r} = \begin{pmatrix} x_P \\ y_P \end{pmatrix} + \lambda \begin{pmatrix} y_B - y_A \\ x_A - x_B \end{pmatrix}$$

Nun muss das λ ermittelt werden, damit die y-Komponente der Gerade genau Null beträgt.

$$y_P + \lambda(x_A - x_B) = 0$$
$$\lambda = \frac{y_P}{x_B - x_A}$$

Wird dieses λ in die Geradengleichung eingesetzt, dann ist die x-Komponente genau x-Koordinate des Mittelpunkt des Kreises.

$$x_M = x_P + \lambda(y_B - y_A)$$

Jetzt, da der Mittelpunkt bekannt ist, kann die Tangente t_A an A gelegt werden. Für den Stützpunkt der Tangente wird der Punkt A gewählt, als Ortsvektor wird der Vektor $\vec{MA}$ benutzt. Analog dazu

wird die Gleichung für die Tangente t_B erstellt.

$$t_A : \vec{r} = \begin{pmatrix} x_A \\ y_A \end{pmatrix} + a \begin{pmatrix} y_A \\ x_M - x_A \end{pmatrix}$$
$$t_B : \vec{r} = \begin{pmatrix} x_B \\ y_B \end{pmatrix} + b \begin{pmatrix} y_B \\ x_M - x_B \end{pmatrix}$$

Der Schnittpunkt der Tangenten kann ermittelt werden, indem die x- und y-Komponenten einander gleichgestellt werden und dann der Wert von mindestens einem der Skalare a und b ermittelt wird. Dies ergibt ein Gleichungsystem mit zwei Unbekannten und zwei Gleichungen:

$$\begin{vmatrix} x_A & + & ay_A & = & x_B & + & by_bB \\ y_A & + & a(x_M - x_A) & = & y_B & + & b(x_M - x_B) \end{vmatrix}$$

Auf die Darstellung des Lösungsweges wird hier verzichtet. Das Resultat für b beträgt:

$$b = \frac{y_A(y_B - y_A) - (x_M - x_A)(x_B - x_A)}{y_B(x_M - x_A) - y_A(x_M - x_B)}$$

Dieses b kann nun in die Gleichung von t_B eingesetzt werden. Dadurch erhält man die Koordinaten des Schnittpunktes.

$$x_S = x_B + by_B \qquad\qquad y_S = y_B + b(x_M - x_B)$$

Für den rechnerischen Teil des Porgrammes werden diese Gleichungen verwendet. Für den zeichnerischen Teil werden zusätzlich noch die Winkel α und β berechnet.

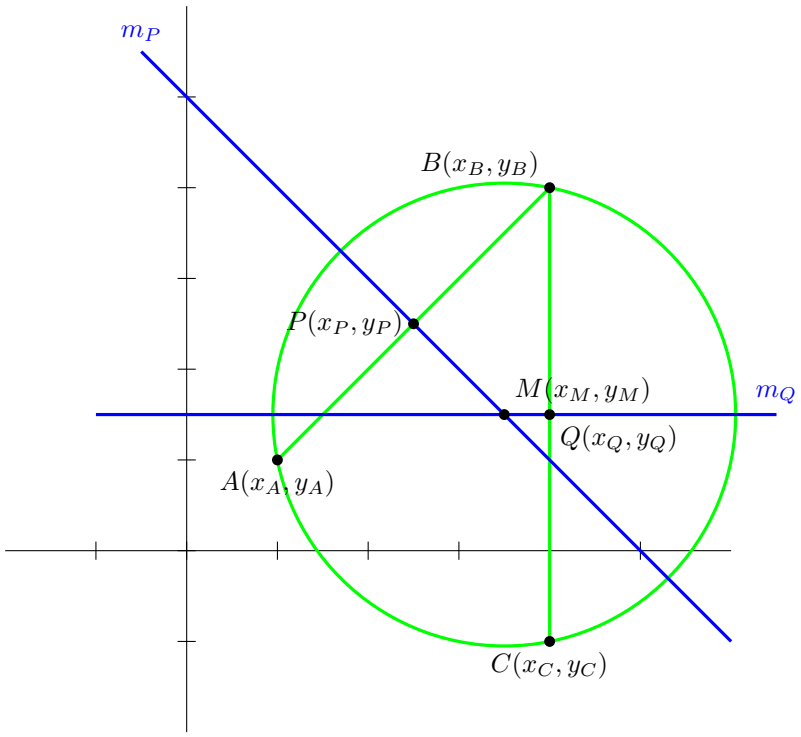
```
to KREISTAN :XA :YA :XB :YB
if :XA=:XB [pr [Aufgabe nicht loesbar]stop]
make "AB sqrt ((:YB-:YA)*(:YB-:YA)+(:XB-:XA)*(:XB-:XA))
make "XP (:XA-:XB)/2+:XB
make "YP (:YA-:YB)/2+:YB
if :XA=:XP [make "XM :XP]
           [make "LAMBDA (-:YP)/(:XA-:XB)
            make "XM :XP+:LAMBDA*(:YB-:YA)]
make "R sqrt ((:XB-:XM)*(:XB-:XM)+:YB*:YB)
if :XM=:XA
[if :YA>0 [make "ALPHA 90]
          [make "ALPHA 270]]
[if :YA<0 [if :XA<:XM [make "ALPHA 180+atan (:YA/(:XA-:XM))]
                      [make "ALPHA atan (:YA/(:XA-:XM))]]
          [if :XA<:XM [make "ALPHA 180+atan (:YA/(:XA-:XM))]
                      [make "ALPHA atan (:YA/(:XA-:XM))]]]
if :XM=:XB
[if :YB>0 [make "BETA 90]
          [make "BETA 270]]
[if :YB<0 [if :XB<:XM [make "BETA 180+atan (:YB/(:XB-:XM))]
                      [make "BETA atan (:YB/(:XB-:XM))]]
          [if :XB<:XM [make "BETA 180+atan (:YB/(:XB-:XM))]
                      [make "BETA atan (:YB/(:XB-:XM))]]]
PUNKT 30 :XA :YA
pu bk :XA*30 lt 90 bk :YA*30 pd
PUNKT 30 :XB :YB
pu bk :XB*30 lt 90 bk :YB*30 pd
```

```
PUNKT 30 :XM 0
KREISMITT :R*30
lt :ALPHA fd :R*30
rt 90 fd 1000 bk 2000 fd 1000 lt 90
bk :R*30 rt :ALPHA
lt :BETA fd :R*30
rt 90 fd 1000 bk 2000 fd 1000 lt 90
bk :R*30 rt :BETA
make "B1 :YA*(:YB-:YA)-(:XM-:XA)*(:XB-:XA)
make "B2 :YB*(:XM-:XA)-:YA*(:XM-:XB)
make "B :B1/:B2
make "XS :XB+:B*:YB
make "YS :YB+:B*(:XM-:XB)
pr [XS=] pr :XS
pr [YS=] pr :YS
end
```

Alle im Programm vorkommenden Strecken, Winkel und Punkte sind in der Skizze zur Aufgabe eingezeichnet.

Die Verzweigungen vor dem Erstellen der des Winkels α sind notwendig, da der Arcustangens immer zwischen -90 und 90 Grad liegt, egal, ob der ursprüngliche Winkel größer ist oder nicht.

Kontrollaufgabe 3



Bevor gezeichnet werden kann werden alle benötigten Werte berechnet. Das sind die Koordinaten des Mittelpunktes, der Radius sowie der Winkel zwischen der Parallelen der x-Achse, die durch den Kreismittelpunkt M läuft, und der Strecke $\overline{MA}$. Dieser Winkel wird benötigt, damit der Radius von M nach A eingezeichnet werden kann.

Um den Mittelpunkt des Kreises zu ermitteln werden die Mittelsenkrechten der Strecken $\overline{AB}$ und $\overline{BC}$ verwendet. Der Mittelpunkt liegt auf dem Schnittpunkt der beiden Mittelsenkrechten. Um den Mittelpunkt rechnerisch zu finden, müssen die Koordinaten der beiden Punkte P und Q ermittelt werden:

$$x_P = \frac{x_A - x_B}{2} + x_B \qquad x_Q = \frac{x_B - x_C}{2} + x_C$$
$$y_P = \frac{y_A - y_B}{2} + y_B \qquad y_Q = \frac{y_B - y_C}{2} + y_C$$

Sind diese Koordinaten bekannt, können die Gleichungen der Geraden erstellt werden. Als Stützpunkt dient P bzw. Q und als Ortsvektor der negative Kehrwert der Strecke $\overline{AB}$ bzw. $\overline{BC}$.

$$m_P : \vec{r} = \begin{pmatrix} x_P \\ y_P \end{pmatrix} + p \begin{pmatrix} y_B - y_A \\ x_A - x_B \end{pmatrix}$$
$$m_Q : \vec{r} = \begin{pmatrix} x_Q \\ y_Q \end{pmatrix} + q \begin{pmatrix} y_C - y_B \\ x_B - x_C \end{pmatrix}$$

Nun müssen die beiden Skalare p und q so gewählt werden, dass die x- und y-Kopmonenten der beiden Geraden gleich gross sind. Wenn dies der Fall ist, dann sind eben diese x- und y-Kopmonenten die Koordinaten des Kreismittelpunktes M.

Um die Werte für p und q zu ermitteln, muss im folgenden Gleichungssystem mindestens der Wert von p oder q ermittelt werden:

$$\begin{vmatrix} x_P + p(y_B - Y_A) & = x_Q + q(y_C - y_B) \\ y_P + p(x_A - x_B) & = y_Q + q(x_B - x_C) \end{vmatrix}$$

Der Lösungsweg der Gleichung wird hier ausgespart. Die Variable q erhält den Wert:

$$q = \frac{(y_B - y_A)(y_Q - y_P) - (x_A - x_B)(x_Q - x_P)}{(x_A - x_B)(y_C - y_B) - (y_B - y_A)(x_B - x_C)}$$

Damit können nun die Koordinaten von M berechnet werden:

$$x_M = x_Q + q(y_C - y_B) \qquad y_M = y_Q + q(x_B - x_C)$$

Der Winkel α wird mit Hilfe des Arcustangens berechnet.

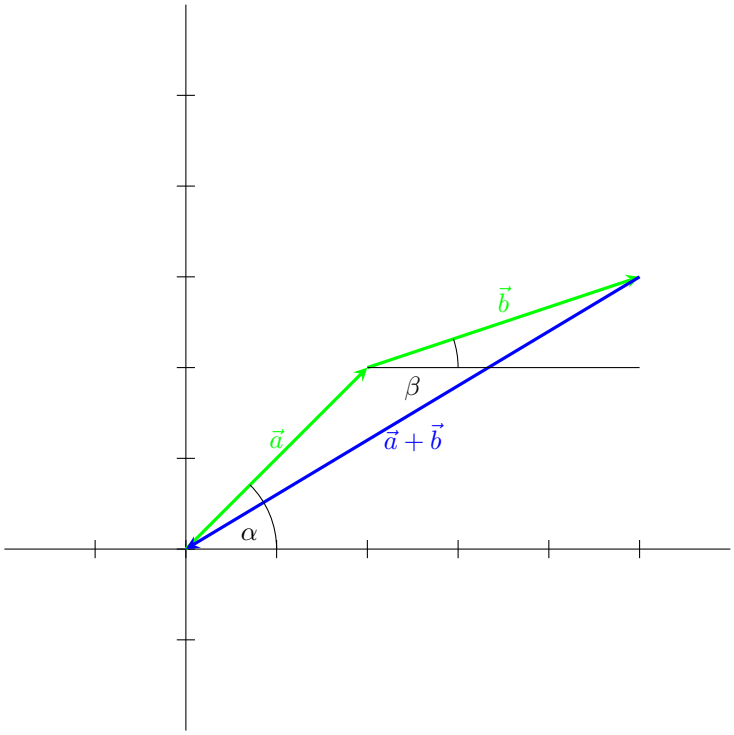
Diese Formeln werden nun im Programmablauf verwendet, um die zum Zeichnen benötigten Werte x_M, y_M, r und α zu berechnen.

```
to KREIS3PUNKT :XA :YA :XB :YB :XC :YC
make "XQ (:XB-:XC)/2+:XC
make "YQ (:YB-:YC)/2+:YC
make "XP (:XA-:XB)/2+:XB
make "YP (:YA-:YB)/2+:YB
make "Q1 (:YB-:YA)*(:YQ-:YP)-(:XA-:XB)*(:XQ-:XP)
make "Q2 (:XA-:XB)*(:YC-:YB)-(:YB-:YA)*(:XB-:XC)
make "Q :Q1/:Q2
make "XM :XQ+:Q*(:YC-:YB)
make "YM :YQ+:Q*(:XB-:XC)
make "R sqrt ((:XA-:XM)*(:XA-:XM)+(:YA-:YM)*(:YA-:YM))
if :XM=:XA
[if :YA>:YM [make "ALPHA 90]
            [make "ALPHA 270]]
[if :YA<:YM [if :XA<:XM [make "ALPHA 180+atan ((:YA-:YM)/(:XA-:XM))]
```

```
                        [make "ALPHA atan ((:YA-:YM)/(:XA-:XM))]]
            [if :XA<:XM [make "ALPHA 180+atan ((:YA-:YM)/(:XA-:XM))]
                        [make "ALPHA atan ((:YA-:YM)/(:XA-:XM))]]]
PUNKT 30 :XA :YA
pu bk :XA*30 lt 90 bk :YA*30 pd
PUNKT 30 :XB :YB
pu bk :XB*30 lt 90 bk :YB*30 pd
PUNKT 30 :XC :YC
pu bk :XC*30 lt 90 bk :YC*30 pd
PUNKT 30 :XM :YM
KREISMITT :R*30
lt :ALPHA fd :R*30 bk :R*30
end
```

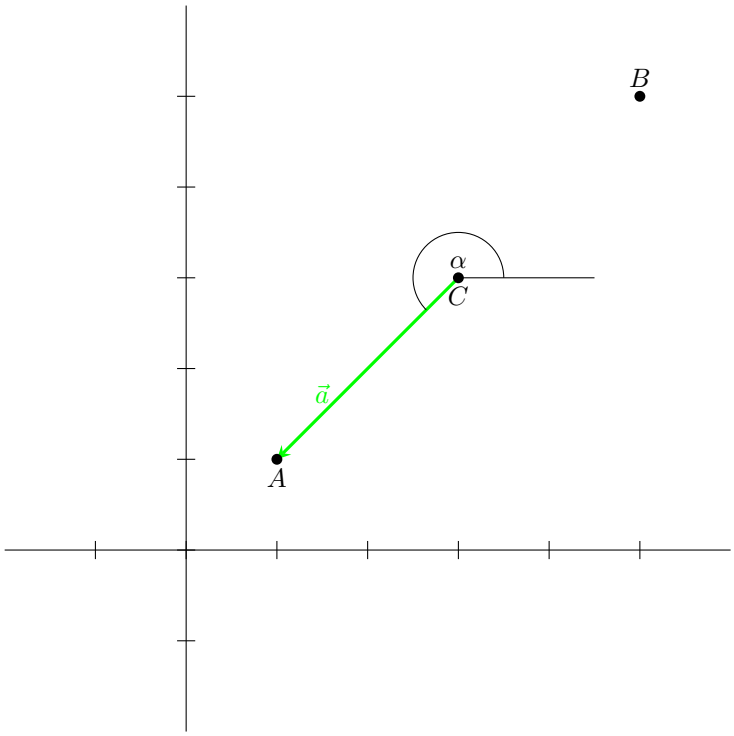
Die Verzweigungen vor dem Erstellen der des Winkels α sind notwendig, da der Arcustangens immer zwischen -90 und 90 Grad liegt, egal, ob der ursprüngliche Winkel größer ist oder nicht.

Kontrollaufgabe 4




```
to VEKTORADD :ALPHA :A BETA :B
KOOR 10 30
setpc [0 255 0] rt 90
lt :ALPHA fd :A*30 rt :ALPHA
lt :BETA fd :B*30 rt :BETA
setpc [0 0 255]
home
end
```

Kontrollaufgabe 5



Um die Zeichnung zu erstellen, müssen die Koordinaten des Punktes C ermittelt werden.

$$x_C = \frac{x_A - x_B}{2} + x_B \qquad\qquad y_C = \frac{y_A - y_B}{2} + y_B$$

Der Winkel α wird für $x_A > x_C$ und $y_A > y_C$ so berechnet:

$$tan(\alpha) = \frac{y_A - y_C}{x_A - x_C}$$

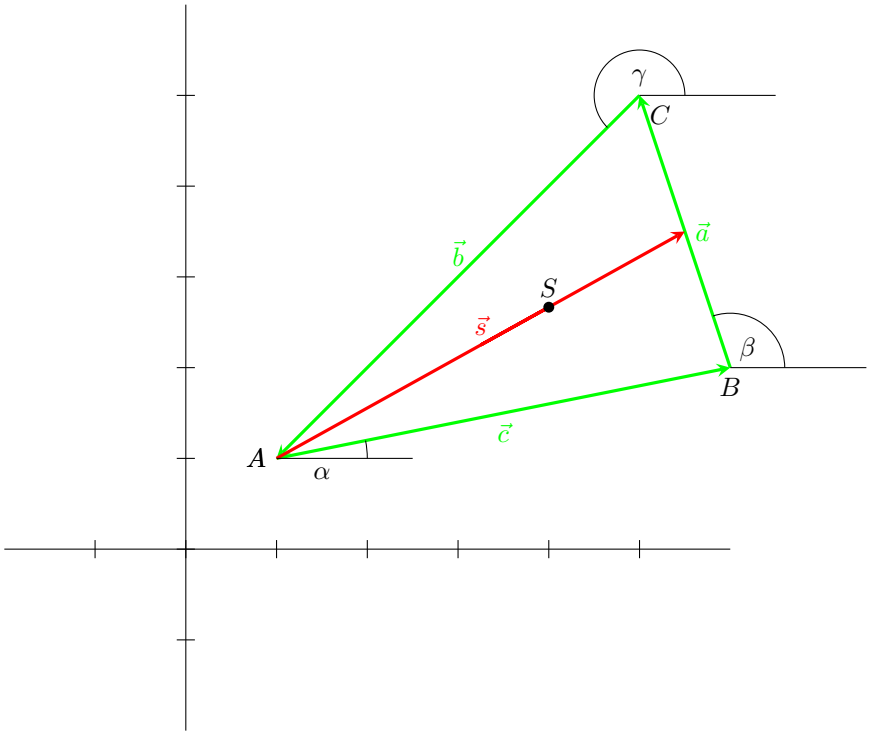
$$\alpha = arctan(\frac{y_A - y_C}{x_A - x_C})$$

Da der Arcustangens immer Winkel zwischen -90 und 90 Grad liefert, muss der Winkel mittels Verzweigungen noch an die tatsächlichen Gegebenheiten angepasst werden.

```
to VEKTORSUB :XA :YA :XB :YB
make "XC (:XA-:XB)/2+:XB
make "YC (:YA-:YB)/2+:YB
make "A sqrt ((:XC-:XA)*(:XC-:XA)+(:YC-:YA)*(:YC-:YA))
if :XC=:XA
[if :YA>:YC [make "ALPHA 90]
            [make "ALPHA 270]]
[if :YA<:YC [if :XA<:XC [make "ALPHA 180+atan ((:YA-:YC)/(:XA-:XC))]
                        [make "ALPHA atan ((:YA-:YC)/(:XA-:XC))]]
            [if :XA<:XC [make "ALPHA 180+atan ((:YA-:YC)/(:XA-:XC))]
                        [make "ALPHA atan ((:YA-:YC)/(:XA-:XC))]]]
PUNKT 30 :XA :YA
pu bk :XA*30 lt 90 bk :YA*30 pd
PUNKT 30 :XB :YB
```

```
pu bk :XB*30 lt 90 bk :YB*30 pd
PUNKT 30 :XC :YC
setpc [0 255 0] lt :ALPHA fd :A*30
end
```

Kontrollaufgabe 6



Zum Zeichen des Dreiecks werden die drei Winkel α, β und γ benötigt. Diese werden mit Hilfe des Arcustangens berechnet. Die Koorodianten des Schwerpunktes S werden wie folgt berechnet:

$$S = A + \vec{c} + \frac{1}{2}\vec{a} - \frac{1}{3}\vec{s}$$

Für $\vec{s}$ kann man $\vec{c} + \frac{1}{2}\vec{a}$ einsetzen.

$$\begin{aligned} S &= A + \vec{c} + \frac{1}{2}\vec{a} - \frac{1}{3}(\vec{c} + \frac{1}{2}\vec{a}) \\ &= A + \vec{c} + \frac{1}{2}\vec{a} - \frac{1}{3}\vec{c} - \frac{1}{6}\vec{a} \\ &= A + \frac{2}{3}\vec{c} + \frac{1}{3}\vec{a} \\ &= \begin{pmatrix} x_A \\ y_A \end{pmatrix} + \frac{2}{3}\begin{pmatrix} x_B - x_A \\ y_B - y_A \end{pmatrix} + \frac{2}{6}\begin{pmatrix} x_C - x_b \\ y_C - y_B \end{pmatrix} \end{aligned}$$

Damit kann man die Koordinaten von S berechenen:

$$\begin{aligned} x_S &= x_A + \frac{2}{3}(x_B - x_A) + \frac{1}{3}(x_C - x_B) \\ &= x_A + \frac{2}{3}x_B - \frac{2}{3}x_A + \frac{1}{3}x_C - \frac{1}{3}x_B \\ &= \frac{1}{3}x_A + \frac{1}{3}x_B + \frac{1}{3}x_C \end{aligned}$$

$$\begin{aligned} y_S &= y_A + \frac{2}{3}(y_B - y_A) + \frac{1}{3}(y_C - y_B) \\ &= y_A + \frac{2}{3}y_B - \frac{2}{3}y_A + \frac{1}{3}y_C - \frac{1}{3}y_B \\ &= \frac{1}{3}y_A + \frac{1}{3}y_B + \frac{1}{3}y_C \end{aligned}$$

Nun wird das Porgramm geschrieben:

```
to DREIECKS :XA :YA :XB :YB :XC :YC
if :XB=:XA
[if :YB>:YA [make "ALPHA 90]
            [make "ALPHA 270]]
[if :YB<:YA [if :XB<:XA [make "ALPHA 180+atan ((:YB-:YA)/(:XB-:XA))]
                        [make "ALPHA atan ((:YB-:YA)/(:XB-:XA))]]
            [if :XB<:XA [make "ALPHA 180+atan ((:YB-:YA)/(:XB-:XA))]
                        [make "ALPHA atan ((:YB-:YA)/(:XB-:XA))]]]
if :XC=:XB
[if :YC>:YB [make "BETA 90]
            [make "BETA 270]]
[if :YC<:YB [if :XB<:XA [make "BETA 180+atan ((:YC-:YB)/(:XC-:XB))]
                        [make "BETA atan ((:YC-:YB)/(:XC-:XB))]]
            [if :XC<:XB [make "BETA 180+atan ((:YC-:YB)/(:XC-:XB))]
                        [make "BETA atan ((:YC-:YB)/(:XC-:XB))]]]
if :XA=:XC
[if :YA>:YC [make "GAMMA 90]
            [make "GAMMA 270]]
[if :YA<:YC [if :XA<:XC [make "GAMMA 180+atan (abs(:YA-:YC)/(:XA-:XC))]
                        [make "GAMMA atan (abs(:YA-:YC)/(:XA-:XC))]]
            [if :XA<:XC [make "GAMMA 180+atan (abs(:YA-:YC)/(:XA-:XC))]
                        [make "GAMMA atan (abs(:YA-:YC)/(:XA-:XC))]]]
make "XS 1/3*:XA+1/3*:XB+1/3*:XC
make "YS 1/3*:YA+1/3*:YB+1/3*:YC
make "A sqrt ((:XC-:XB)*(:XC-:XB)+(:YC-:YB)*(:YC-:YB))
make "B sqrt ((:XA-:XC)*(:XA-:XC)+(:YA-:YC)*(:YA-:YC))
make "C sqrt ((:XB-:XA)*(:XB-:XA)+(:YB-:YA)*(:YB-:YA))
PUNKT 30 :XA :YA
setpc [0 255 0]
lt :ALPHA fd :C*30 bk :C*30 rt :ALPHA
pu bk :XA*30 lt 90 bk :YA*30 pd
setpc [0 0 0]
PUNKT 30 :XB :YB
setpc [0 255 0]
lt :BETA fd :A*30 bk :A*30 rt :BETA
pu bk :XB*30 lt 90 bk :YB*30 pd
setpc [0 0 0]
PUNKT 30 :XC :YC
setpc [0 255 0]
lt :GAMMA fd :B*30 bk :B*30 rt :GAMMA
pu bk :XC*30 lt 90 bk :YC*30 pd
setpc [0 0 0]
PUNKT 30 :XS :YS
end
```

Da der Arcustangens immer Winkel zwischen -90 und 90 Grad liefert, muss der Winkel mittels Verzweigungen noch an die tatsächlichen Gegebenheiten angepasst werden.

Kontrollaufgabe 7 Zuerst werden die Richtungswinkel der beiden Vektoren mittels Arcustangens ermittelt. Dann wird um den ersten Winkel gedreht und um die Distanz `D1*EINHEIT` vorwärts gezeichnet. Dasselbe wird an der Spitze des nun gezeichnet Vektors wiederholt, einfach mit der Länge $D_1 \cdot$ Einheit.

```
to VEKTORDIST :X1 :Y1 :X2 :Y2 :D1 :D2
KOOR 10 30
if :X1=0
[if :Y1>0 [make "ALPHA 90]
          [make "ALPHA 270]]
[if :Y1<0 [if :X1<0 [make "ALPHA 180+atan ((:Y1)/(:X1))]
                    [make "ALPHA atan ((:Y1)/(:X1))]]
          [if :X1<0 [make "ALPHA 180+atan ((:Y1)/(:X1))]
                    [make "ALPHA atan ((:Y1)/(:X1))]]]
if :X2=0
[if :Y2>0 [make "BETA 90]
          [make "BETA 270]]
[if :Y2<0 [if :X2<0 [make "BETA 180+atan ((:Y2)/(:X2))]
                    [make "BETA atan ((:Y2)/(:X2))]]
          [if :X2<0 [make "BETA 180+atan ((:Y2)/(:X2))]
                    [make "BETA atan ((:Y2)/(:X2))]]]
setpc [0 255 0]
rt 90-:ALPHA fd :D1*30 rt :ALPHA
lt :BETA fd :D2*30
end
```

Da der Arcustangens immer Winkel zwischen -90 und 90 Grad liefert, muss der Winkel mittels Verzweigungen noch an die tatsächlichen Gegebenheiten angepasst werden.

Kontrollaufgabe 8

```
to VIERECKD :XA :YA :XB :YB :XC :YC :XD :YD
```