Lektion 5

Programme mit Parametern

Die Geschichte mit der „Faulheit“ der Programmierer nimmt kein Ende. In Lektion 3 haben wir aus diesem Grund gelernt, Programmen einen Namen zu geben und sie dann mit diesem Namen aufzurufen, um das gewünschte Bild vom Rechner zeichnen zu lassen. Das bedeutet, dass der Name des von uns benannten Programms zu einem Befehlsnamen wurde. Und weil wir bei diesem Befehlsnamen, wie z. B. `QUAD100`, keine Parameter verwendet haben, ist dieser Befehlsname zu einem eigenständigen Befehl geworden wie `cs` oder `pu`. Wenn wir mehrfach das gleiche Bild zeichnen wollen, z. B. `SCHW100`, dann spart uns die Verwendung des Programmnamens als neuer Befehl viel Tippen, weil wir sonst wiederholt das gleiche Programm eintippen müssten.

Es gibt aber auch Situationen, in denen uns dieses Vorgehen nicht hinreichend hilft. Nehmen wir an, wir wollen fünf Quadrate unterschiedlicher Größe zeichnen. Dabei sollen die Seitenlängen jeweils 50, 70, 90, 110 und 130 sein und es soll die Zeichnung in Abb. 5.1 auf der nächsten Seite entstehen. Stellen wir uns weiter vor, dass wir häufiger Quadrate dieser Größe zeichnen müssen.

Wir könnten so vorgehen, dass wir fünf Programme schreiben, jeweils ein Programm für die Quadratgröße 50, 70, 90, 110 und 130. Das würde dann so aussehen:

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

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

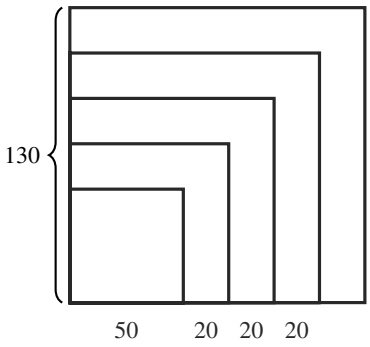


Abbildung 5.1

```
to QUAD 90
repeat 4 [ fd 90 rt 90 ]
end
```

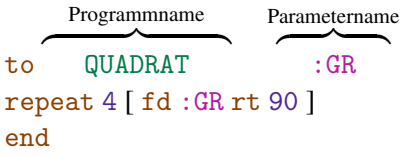
Und so weiter. Das sieht langweilig aus, oder? Wir schreiben immer fast das gleiche Programm. Der Unterschied liegt nur in der Zahl mit gelbem Hintergrund, die der Seitenlänge des jeweiligen Quadrats entspricht. Wäre es nicht schöner und praktischer einen Befehl

```
QUADRAT X
```

mit einem Parameter `X` zu haben, der für eine Zahl `X` das Quadrat mit Seitenlänge `X` zeichnet? Das wäre nach dem Muster von `fd X`. Der Befehl `fd X` besteht aus einem Befehlswort `fd` und einem Parameter `X` und zeichnet eine Linie der Länge `X`. Warum sollte es nicht möglich sein, einen Befehl mit Parameter für das Zeichnen von Quadraten beliebiger Größe zu haben?

Und dies geht tatsächlich mit dem Konzept der **Programmparameter**, die wir meistens nur kurz als **Parameter** bezeichnen werden. Bei der Beschreibung eines Programms mit einem Parameter sucht man sich für den Parameter zuerst einen Namen. Dann schreibt man den Befehl `to`, dahinter den `PROGRAMMNAMEN` und dahinter den `:PARAMETERNAMEN`. Vor dem Parameternamen muss immer ein Doppelpunkt stehen. Dadurch erkennt der Rechner, dass es sich um einen Parameter handelt. Danach folgt ein fast gleiches Programm wie das, was wir ohne Parameter hatten. Nur dass man dort, wo sich die Zahlen abhängig von der Bildgröße unterscheiden (in unserem Beispiel die Zahlen mit gelbem Hintergrund), statt der Zahlen den Parameternamen (mit Doppelpunkt davor) schreibt.

Somit sieht das Programm zum Zeichnen von Quadraten in beliebiger Größe folgendermaßen aus:



Beachte, dass `:GR` genau an der Stelle vorkommt, wo in den Programmen `QUAD50`, `QUAD70` und `QUAD90` die jeweiligen Seitenlängen 50, 70 bzw. 90 standen.

Hinweis für die Lehrperson Eine der schwersten Hürden des Programmierunterrichts ist das Konzept der Variable. Parameter sind nichts anderes als „passive“ Variablen, also Variablen, deren Wert sich während der Ausführung des entsprechenden Programmaufrufs nicht ändert. Deswegen ist die Einführung des Parameters eine wichtige didaktische Vorgehensweise auf dem Weg zur Bewältigung des Konzepts von Variablen. Unsere Unterrichtsexperimente zeigen, dass man das Konzept von Parametern ab dem Alter von 9 Jahren erfolgreich unterrichten kann, obwohl das Konzept der Variablen frühestens ab dem 6. Schuljahr für große Klassen zugänglich ist. Da mit der Einführung von Programmparametern die Anforderungen im Programmierunterricht wesentlich steigen, empfehlt es sich, in dieser Lektion konsequent alle Aufgaben durch zuarbeiten.

Um die Schreibweise von Programmen mit Parametern zu verstehen, erklären wir zuerst, was in einem Rechner passiert, wenn man dieses Programm eintippt. Wie wir schon wissen, wird das Programm unter dem Namen `QUADRAT` abgespeichert. Zusätzlich wird notiert, dass es einen Parameter hat. Wenn der Rechner einen Doppelpunkt liest, weiß er, dass danach der Parametername folgt. Wenn also der Rechner die erste Zeile

```
to QUADRAT :GR
```

liest, reserviert er einen Speicherplatz, genannt Register, in seinem Speicher und gibt ihm den Namen `:GR`. Das Ganze sieht aus wie in Abb. 5.2(a).

Den Speicher kann man sich als Schrank mit vielen Schubladen vorstellen. Die Schubladen sind die kleinste Einheit des Schranks und heißen **Register**. In jedem Register kann man genau eine Zahl abspeichern und jedem Register kann man einen Namen geben. Wenn man in ein Register keine Zahl gespeichert hat, liegt da automatisch eine Null, damit ist ein Register nie leer. Die Zahl, die in einem Register R gespeichert ist, bezeichnen wir als den Inhalt des Registers R. In der Abb. 5.2(a) hat der Rechner dem

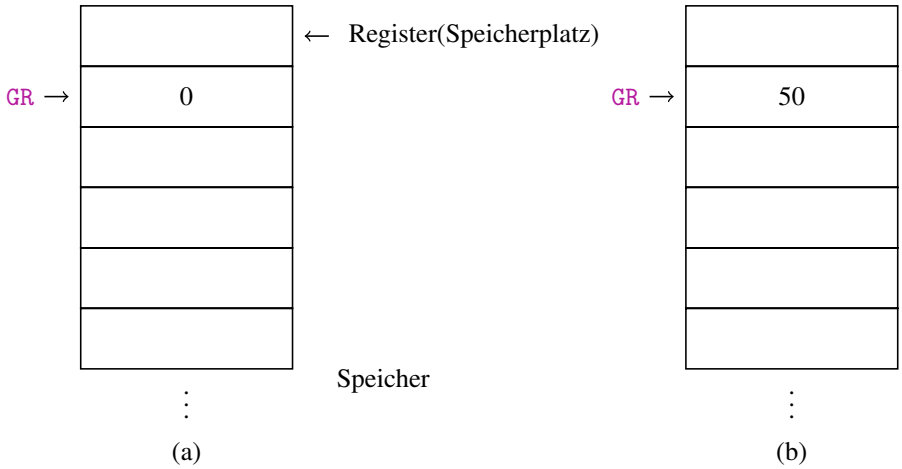


Abbildung 5.2 Schematischer Aufbau des Rechnerspeichers, der aus einer großen Anzahl von Registern besteht. Register sind die kleinsten Einheiten des Speichers, die man ansprechen kann.

zweiten Register den Namen GR gegeben und weil wir noch nichts „rein gelegt“ haben, liegt dort die Zahl 0.

Wenn man jetzt den Befehl

```
QUADRAT 50
```

eintippt, weiß der Rechner, dass `QUADRAT` ein Programm mit dem Parameter `:GR` ist. Die Zahl `50` hinter dem Programmnamen nimmt er dann und legt sie in das Register (in die Schublade) mit dem Namen `:GR`. Wenn sich dort schon eine Zahl befindet, wird sie gelöscht (und damit vergessen) und durch die neue Zahl `50` ersetzt. Die Situation im Speicher nach der Umsetzung des Befehls `QUADRAT 50` ist in Abb. 5.2(b) dargestellt. Danach setzt der Rechner alle Befehle des Programms `QUADRAT` nacheinander um. Überall, wo `:GR` auftritt, geht er an den Speicher, öffnet die Schublade mit dem Namen `GR` und liest den Inhalt des Registers. Mit dem Inhalt, den er dort findet, ersetzt er im Programm die Stelle, die mit `:GR` beschriftet ist und führt den entsprechenden Befehl aus.

Wenn wir im Programm `QUADRAT :GR` alle Stellen, wo `:GR` steht, durch `50` ersetzen, erhalten wir genau das Programm `QUAD50`. Somit zeichnet der Rechner nach dem Befehl `QUADRAT 50` ein Quadrat mit Seitenlänge 50.

Aufgabe 5.1 Nachdem du das Programm `QUADRAT :GR` definiert hast, gib die Befehle

```
QUADRAT 50 QUADRAT 70 QUADRAT 90
QUADRAT 110 QUADRAT 130
```

ein und schaue ob Du dadurch das Bild aus Abb. 5.1 auf Seite 86 gezeichnet hast. Welche Zahl liegt im Register `GR` nach dem Befehl `QUADRAT 70`? Welche Zahl liegt im Register `GR` nach der Durchführung dieses Programms?

Hinweis für die Lehrperson Achten Sie bitte darauf, dass man sich für unterschiedliche Programme unterschiedliche Namen aussucht. Das Gleiche gilt auch für die Parameter.
Es ist wichtig zu betonen, dass in jedem Register immer genau eine Zahl liegt. Es können dort nicht zwei Zahlen abgespeichert werden. Wenn man durch `Befehl Zahl` eine Zahl in ein Register legt, in dem schon eine Zahl gespeichert war, wird die alte Zahl gelöscht und durch die neue ersetzt. Es ist wie ein kleines Band, auf das man nur eine Zahl schreiben darf. Wenn man dort eine andere Zahl aufschreiben will, muss man zuerst die dort stehende Zahl ausradieren. Es ist sehr wichtig, dieses Konzept zu verstehen, da es die Grundlage zur Einführung des Konzepts der Variablen in Lektion 8 ist.

Aufgabe 5.2 Nutze das Programm `QUADRAT :GR`, um Quadrate mit Seitenlänge 50, 99, 123 und 177 zu zeichnen.

Aufgabe 5.3 Was zeichnet das folgende Programm?

```
rt 45
repeat 4 [ QUADRAT 50 QUADRAT 70 QUADRAT 90 QUADRAT 110 rt 90 ]
```

Jemand will das gleiche Bild erzeugen und fängt wie folgt an:

```
rt 45
repeat 4 [ QUADRAT 50 rt 90 ]
```

Kannst du ihm helfen, das Programm zu Ende zu schreiben?

Aufgabe 5.4 Schreibe ein Programm mit einem Parameter, das regelmäßige Sechsecke beliebiger Seitengröße zeichnet. Probiere das Programm zum Zeichnen von Sechsecken für die Seitenlänge 40, 60 und 80 aus.

Aufgabe 5.5 Schreibe ein Programm mit einem Parameter zum Zeichnen von gleichseitigen Fünfecken mit beliebiger Seitenlänge.

Beispiel 5.1 In Lektion 4 haben wir gelernt, gleichseitige Vielecke mit beliebig vielen Ecken zu zeichnen. Jetzt wollen wir ein Programm entwickeln, das Vielecke mit beliebig vielen Ecken und einer Seitenlänge von 50 zeichnet. Schauen wir uns zuerst die Beispiele der folgenden Programme an, die jeweils ein 7-Eck, ein 12-Eck und ein 18-Eck zeichnen.

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

repeat 12 [ fd 50 rt 360/ 12 ]

repeat 18 [ fd 50 rt 360/ 18 ]
```

Wie beim Zeichnen von Quadraten unterschiedlicher Größe sind die Programme bis auf die Stellen mit gelbem Hintergrund gleich. Der einzige Unterschied zur vorherigen Aufgabe ist, dass der Parameter des Programms an zwei unterschiedlichen Stellen im Programm auftritt. Dementsprechend taucht der Parametername in der Beschreibung des Programms zweimal auf.

```
to VIELECK :ECKE
repeat :ECKE [ fd 50 rt 360/:ECKE ]
end
```

Bei dem Aufruf `VIELECK 7` wird die Zahl 7 im Register `ECKE` gespeichert. Bei der Ausführung des Programms werden die beiden Stellen im Programm mit `:ECKE` durch die Zahl 7 (dem Inhalt des Registers `ECKE`) ersetzt. □

Hinweis für die Lehrperson In den ersten Lektionen verwenden wir konsequent große Buchstaben für die Parameternamen. Beide Programmiersprachen XLOGO und SUPERLOGO erlauben auch die Verwendung von kleinen Buchstaben. Hier muss man aber vorsichtig sein. Die Programme unterscheiden nicht zwischen großen und kleinen Buchstaben. Die Parameternamen `:a` und `:A` haben somit die gleiche Bedeutung und werden nicht als zwei unterschiedliche Namen behandelt.

Aufgabe 5.6 Verwende das Programm `VIELECK` und zeichne hintereinander fünf regelmäßige Vielecke mit einer unterschiedlichen Anzahl an Ecken bei einer Seitenlänge von 50.

Aufgabe 5.7 Schreibe ein Programm mit einem Parameter, das unterschiedlich große Kreise zeichnen kann.

Aufgabe 5.8 In Lektion 3 haben wir gelernt, eine fette Linie zu zeichnen. Schreibe ein Programm zum Zeichnen einer fetten Linie (als Doppellinie) mit beliebiger Länge.

Aufgabe 5.9 Schreibe ein Programm mit einem Parameter `:X`, das beliebig große Häuser wie in Abb. 5.3 zeichnet.

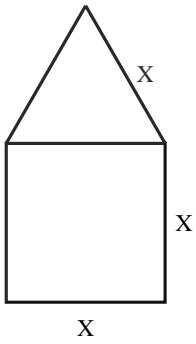


Abbildung 5.3

Schaffst du es auch, ein beliebig großes Haus wie in Abb. 5.4 mit Hilfe nur eines Parameters zeichnen zu lassen?

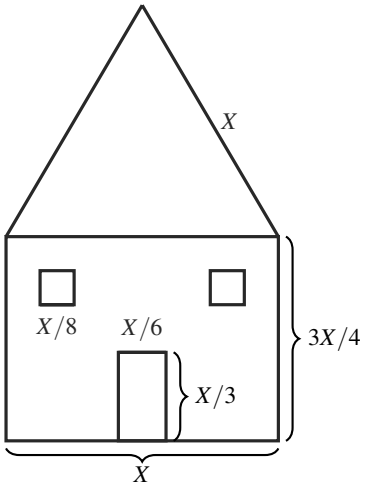


Abbildung 5.4

Beide Fenster liegen in der Entfernung $X/8$ von der seitlichen Hauswand und in der Entfernung $X/8$ vom Dachgeschoss.

Die Parameter ermöglichen es, einige Eigenschaften der Bilder frei wählbar zu lassen. Wir müssen also beim Schreiben des Programms noch nicht bestimmen, wie groß das Bild sein soll. Wir können die Größe später bei der Ausführung wählen. Aber es geht nicht nur um die Seitenlänge oder andere Größen. Man kann auch Parameter nutzen, um die Anzahl an Wiederholungen einer Zeichnung frei wählbar zu halten. Zum Beispiel in Abb. 2.6 auf Seite 44 haben wir zehn Quadrate der Seitenlänge 20 gezeichnet. Wenn wir aber die Zahl der Quadrate, die gezeichnet werden sollen, frei wählen wollen, dann ersetzen wir im Programm aus Beispiel 2.3 (erste Lösung) die Zahl 10 durch einen Parameter `:ANZ`.

```
to LEITER :ANZ
repeat :ANZ [ repeat 4 [ fd 20 rt 90 ] rt 90 fd 20 lt 90 ]
end
```

Aufgabe 5.10 Schreibe Programme, die eine frei wählbare Anzahl von Treppenstufen wie in Abb. 2.1 auf Seite 39 und Abb. 2.2 auf Seite 40 zeichnen.

Aufgabe 5.11 Schreibe ein Programm, das eine frei wählbare Anzahl von Radzähnen (Abb. 2.5 auf Seite 44) zeichnet.

Beispiel 5.2 Wir wollen zuerst ein Programm `KREISW :UM` entwerfen, dass für einen gegebenen Wert für `UM` einen Kreis vom Umfang `UM` zeichnet. Dafür betrachten wir das Programm

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

Der Umfang des gezeichneten Kreises, welches eigentlich ein 360-Eck ist, kann durch die Anzahl der Seiten (360) mal die Seitengröße (`UM`/`360`) berechnet werden und ist somit tatsächlich `UM`.

Jetzt wollen wir ein Programm für die Zeichnung in Abbildung 5.5 auf der nächsten Seite entwerfen. Dabei soll der Umfang des großen Kreises frei wählbar sein und die zwei kleineren Kreise sollten den halben Umfang haben. Es gibt mehrere Möglichkeiten, wie wir vorgehen können. Wir zeichnen zuerst einen großen Halbkreis, danach den rechten kleinen Kreis, dann vervollständigen wir den großen Kreis und letztendlich zeichnen wir den linken kleinen Kreis.

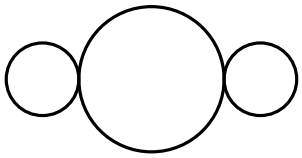


Abbildung 5.5

```
to KOPF :UMFANG
repeat 180 [ fd :UMFANG/360 rt 1 ]
repeat 360 [ fd :UMFANG/720 lt 1 ]
repeat 180 [ fd :UMFANG/360 rt 1 ]
repeat 360 [ fd :UMFANG/720 lt 1 ]
end
```

Aufgabe 5.12 Zeichne das Bild in Abb. 5.6. Die Größe des Umfangs des größten Kreises soll frei wählbar sein, die kleineren Kreise sollen einen dreimal kleineren Umfang haben als der große Kreis.

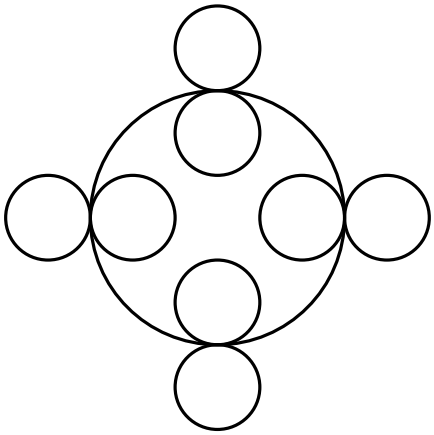
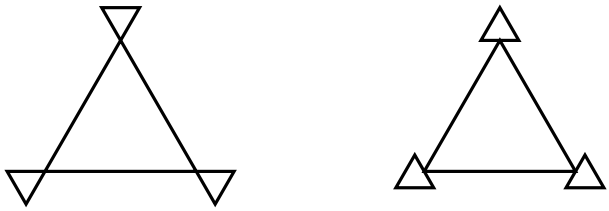


Abbildung 5.6

Aufgabe 5.13 Zeichne die Bilder in Abb. 5.7 (a) und (b). Die Seitengröße des größten Dreiecks sollte frei wählbar sein und die kleinen Dreiecke sollten viermal kleiner werden als das große Dreieck.

Bisher haben wir nur Programme mit einem Parameter betrachtet. Dadurch war immer nur eine Eigenschaft (Größe, Anzahl der Wiederholungen, . . .) der Bilder frei wählbar. Es



(a) (b)

Abbildung 5.7

kann aber wünschenswert sein, zwei, drei oder mehr Parameter eines Bildes frei wählbar zu haben. Fangen wir mit einem einfachen Beispiel an.

Beispiel 5.3 Wir wollen ein Rechteck mit beliebiger Länge und Breite zeichnen. Wir wählen den Parameter :HOR für die Länge der horizontalen Seite und den Parameter :VER für die Länge der vertikalen Seite. Das Programm

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

zeichnet ein Rechteck der Größe 50×150 und das Programm

```
repeat 2 [ fd 100 rt 90 fd 70 rt 90 ]
```

zeichnet ein Rechteck der Größe 100×70.

Wir sehen sofort, wo die frei wählbaren Größen in den Programmen liegen und so können wir direkt das Programm zum Zeichnen beliebiger Rechtecke aufschreiben.

```
to RECHT :HOR :VER
repeat 2 [ fd :VER rt 90 fd :HOR rt 90 ]
end
```

□

Aufgabe 5.14 Entwerfe ein Programm mit zwei Parametern, um Bilder zu zeichnen wie diejenigen in Abb. 5.5 auf der vorherigen Seite. Dabei sollen die Umfänge beider Kreise frei wählbar sein.

Aufgabe 5.15 Entwerfe ein Programm zum Zeichnen von Bildern wie in Abb. 5.6 auf der vorherigen Seite, wobei der Umfang des großen Kreises sowie der kleinen Kreise frei wählbar ist.

Aufgabe 5.16 Entwerfe ein Programm um Bilder wie in Abb. 5.7 auf der vorherigen Seite zu zeichnen. Beide Größen der Dreiecke sollten mittels zwei Parametern frei wählbar sein. Schaffst du es ein Programm zu schreiben, in dem man die Seitenlänge jedes der vier Dreiecke unabhängig voneinander wählen darf?

Aufgabe 5.17 Schreibe ein Programm zum Zeichnen eines beliebigen roten (rot ausgefüllten) Rechtecks. Dabei sollen beide Seitenlängen frei wählbar sein.

Aufgabe 5.18 Das folgende Programm zeichnet das Parallelogramm aus Abb. 5.8.

```
rt 45 fd 200 rt 45 fd 100 rt 90
rt 45 fd 200 rt 45 fd 100
```

Schreibe ein Programm mit zwei Parametern, das solche Parallelogramme mit beliebigen Seitenlängen zeichnet.

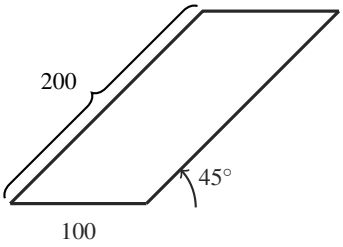


Abbildung 5.8

Beispiel 5.4 Man wünscht sich ein Programm zum Zeichnen beliebiger $X \times Y$-Felder mit wählbarer Seitengröße *GR* der einzelnen Quadrate. In Abb. 2.16 auf Seite 50 ist z. B. $GR = 20$, $X = 4$ und $Y = 10$. Wir können wie folgt vorgehen:

```
to FELD :X :Y :GR
repeat :X [ repeat :Y [ repeat 3 [ fd :GR rt 90 ] rt 90 ] lt 90
            fd :GR*:Y rt 90 fd :GR]
end
```

□

Neu für uns in diesem Programm ist der **arithmetische Ausdruck** `:GR*:Y` im Befehl `fd :GR*:Y`. Hier ersetzt der Rechner erst die Parameternamen `:GR` und `:Y` durch die in

den Registern `GR` und `Y` gespeicherten Zahlen und dann rechnet er das Produkt dieser Zahlen aus. Das Resultat dieser Multiplikation wird nirgendwo gespeichert, sondern nur zur Durchführung des Befehls `fd` verwendet.

Aufgabe 5.19 Teste das Programm aus Beispiel 5.4 und erkläre, wie es funktioniert.

Aufgabe 5.20 Schreibe ein Programm zum Zeichnen beliebiger Rechtecke in beliebiger Farbe.

Aufgabe 5.21 Schreibe ein Programm zum Zeichnen einer frei wählbaren Anzahl Türme wie in Abb. 5.9. Dabei sollen die Höhe und die Breite (oder auch andere Eigenschaften) des Turms frei wählbar sein.

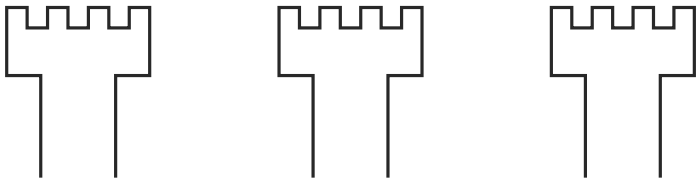


Abbildung 5.9

Aufgabe 5.22 Schreibe ein Programm zum Zeichnen von vier Kreisen wie in Abb. 5.10. Dabei soll die Größe von allen vier Kreisen frei wählbar sein und für jeden der Kreise sollte auch die Farbe frei wählbar sein.

Abbildung 5.10

Zusammenfassung

Programmparameter ermöglichen uns, Programme zu schreiben, die statt einem einzigen Bild eine ganze Klasse von Bildern zeichnen können. Zum Beispiel ein Programm mit drei Parametern reicht zum Zeichnen eines beliebigen Rechtecks in beliebiger Farbe.

Genauso wie Programme muss man Programme mit Parametern mittels eines Befehls `to` definieren. Nach `to` kommt wie üblich der Name des Programms und danach der Name des Parameters (oder der Parameter, falls mehrere vorhanden sind), vor dem immer ein Doppelpunkt geschrieben wird. Auch im Körper des Programms kommt bei jeder Verwendung eines Parameters vor dem Parameternamen ein Doppelpunkt. Der Doppelpunkt signalisiert dem Rechner, dass das, was danach kommt, ein Parameter ist. Wenn wir ein Programm durch seinen Namen aufrufen und dabei anstelle der Parameter konkrete Zahlen eingeben, werden alle Parameternamen im Programm durch die entsprechenden Zahlen ersetzt und das Programm wird mit diesen konkreten Zahlen ausgeführt.

Ein Programm kann eine beliebige Anzahl an Parametern haben. Durch die Parameter können die Größen unterschiedlicher Teile des Bildes, die Anzahl der Wiederholungen von Unterprogrammen oder die Farbe der Bilder gewählt werden.

Wenn man die Definition (Beschreibung) eines Programms mit einem Parameter eintippt, wird das Programm vom Rechner gespeichert und für jeden Parameter wird ein Speicherplatz mit dem Namen des Parameters versehen. Dieser Speicherplatz, welcher Register genannt wird, ist somit für den Parameter reserviert. Beim Aufruf des Programms mit konkreten Zahlen (Parameterwerten) werden automatisch diese Zahlen in den Registern mit den entsprechenden Namen gespeichert.

Die Parameter dürfen arithmetische Ausdrücke als Parameter eines Befehls bilden. Zum Beispiel bei der Verwendung des Befehls `fd :X * :Y - :Z` ersetzt der Rechner die Parameternamen `:X`, `:Y` und `:Z` durch die entsprechenden Zahlen, rechnet das Resultat von $:X \cdot :Y - :Z$ aus und geht dem Resultat entsprechend viele Schritte mit der Schildkröte vorwärts.

Kontrollfragen

1. Welches Zeichen muss immer vor den Namen eines Parameters geschrieben werden?

2. Wie unterscheidet sich die Definition eines einfachen Programms von der Definition eines Programms mit Parametern? Beide starten mit `to` und enden mit `end`.

3. Was sind die Vorteile von Programmen mit Parametern? Was können Programme mit Parametern, was die Programme ohne Parameter nicht können?

4. Was passiert im Speicher des Rechners, nachdem er den Befehl `to NAME :PARAMETER` gelesen hat? Was passiert im Speicher beim Aufruf `NAME 7`?

5. Was kann der Rechner in einem Register speichern? Was passiert, wenn man in einem Register eine Zahl speichern will, es aber nicht leer ist, weil dort früher schon eine Zahl gespeichert worden ist?

6. Wie setzt der Rechner einen Befehl mit Parameter (wie z. B. `fd`, `rt` oder `repeat`) um, wenn statt einer Zahl ein arithmetischer Ausdruck als Parameter dahinter steht?

Kontrollaufgaben

1. Schreibe ein Programm zum Zeichnen der Bilder in Abb. 7 auf Seite 32, bei dem die Größe des Bildes frei wählbar ist.

2. Schreibe ein Programm mit Parametern, das ein $2i \times 2i$-Schachfeld für eine beliebige positive ganze Zahl i zeichnen kann. Dabei soll die Größe der einzelnen Quadrate 50×50 sein.

3. Schreibe ein Programm mit vier Parametern zum Zeichnen eines Baums wie in Abb. 5.11. Die Größe der drei Paare von Zweigen sowie die Farbe des Baums sollen frei wählbar sein.

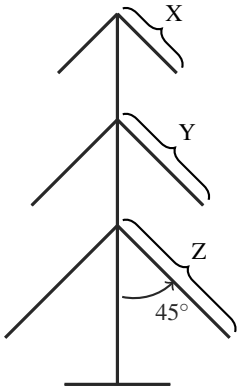


Abbildung 5.11

4. Schreibe ein Programm mit zwei Parametern `:P1` und `:P2`. Das Programm soll ein Rechteck der Länge $P1 \times P2$ und der Breite $2 \times P1$ zeichnen.

5. Schreibe ein Programm `QQQ` mit den drei Parametern `:ANZAHL`, `:LANG` und `:WINK`, das `:ANZAHL` Quadrate der Seitenlänge `:LANG` zeichnet und sich zwischen der Zeichnung zweier Quadrate immer um `:WINK` Grad nach rechts dreht. Teste das Programm mit dem Aufruf `QQQ 23 100 10`.

6. Erweitere das Programm aus Kontrollaufgabe 5 um einen weiteren Parameter `:VOR`. Nach der Drehung um `:WINK` viele Grade gehe `:VOR` viele Schritte mit der Schildkröte nach vorne. Teste das entstandene Programm mit dem Aufruf `QQQ 23 100 10 25`.

Lösungen zu ausgesuchten Aufgaben

Aufgabe 5.4
Das Programm

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

zum Zeichnen eines regelmäßigen 6-Ecks mit Seitenlänge 50 kennen wir schon. Jetzt ersetzen wir die konkrete Seitenlänge 50 durch den Parameter `:L` für die wählbare Seitenlänge und erhalten das folgende Programm:

```
to SECHS :L
repeat 6 [ fd :L rt 60 ]
end
```

Aufgabe 5.7
Wir zeichnen die Kreise als regelmäßige Vielecke mit vielen Ecken. Der Umfang des Kreises ist dann das Produkt der Anzahl der Ecken und der Seitenlänge. Wenn wir verabreden, dass wir die Kreise als 360-Ecke zeichnen, wird die Größe (im Sinne des Umfangs) nur durch die frei wählbare Seitenlänge bestimmt. Also reicht uns ein Parameter `:LA`.

```
to KREISE :LA
repeat 360 [ fd :LA rt 1 ]
end
```

Hier empfehlen wir, beim Aufruf des Befehls `KREISE` auch Kommastellen (`KREISE 1.5`) oder Brüche (`KREISE 7/3`) zu verwenden, um eine größere Vielfalt zu erhalten. Beachte, dass schon `KREISE 3` einen zu großen Kreis zeichnet, der nicht mehr auf unseren Bildschirm passt.

Aufgabe 5.18
Bezeichnen wir die Seitenlängen eines Parallelogramms mit A und B.

```
to PARAL :A :B
rt 45 fd :A rt 45 fd :B rt 90
rt 45 fd :A rt 45 fd :B
end
```

Aufgabe 5.22
Wir brauchen vier Parameter :G1, :G2, :G3 und :G4 für die Größe der Kreise und vier Parameter :F1, :F2, :F3 und :F4 für eine freie Wählbarkeit der Farben.

```
to KREISE4 :G1 :G2 :G3 :G4 :F1 :F2 :F3 :F4
setpencolor :F1
repeat 360 [ fd :G1 rt 1 ]
setpencolor :F2
repeat 360 [ fd :G2 rt 1 ]
setpencolor :F3
repeat 360 [ fd :G3 rt 1 ]
setpencolor :F4
repeat 360 [ fd :G4 rt 1 ]
end
```