Lektion 6

Übergabe von Parameterwerten an Unterprogramme

In Lektion 5 haben wir Programme mit Parametern kennen gelernt und uns von der Nützlichkeit überzeugt. In Lektion 3 haben wir das Konzept des modularen Programmentwurfs behandelt, bei dem wir Programme schreiben, benennen und dann als Bausteine (oder als Befehle) für den Entwurf von komplexeren Programmen verwenden.

Man kann zum Beispiel das Programm `QUAD100` zum Zeichnen eines 100×100-Quadrats verwenden, um das folgende Programm `MUS1` zu schreiben.

```
to MUS1
repeat 20 [ QUAD100 fd 30 rt 10 ]
end
```

Das Programm `QUAD100` nennen wir Unterprogramm des Programms `MUS1`. Das Programm `MUS1` nennen wir in diesem Fall auch Hauptprogramm.

Aufgabe 6.1 Tippe das Programm oben ab und lasse die Schildkröte danach das Muster zeichnen. Wie sieht das Bild aus, wenn man statt `fd 30` im Programm den Befehl `fd 80` oder `fd 50` verwendet?

Der modulare Entwurf von Programmen ist für uns wichtig und wir wollen es auch gerne für Programme mit Parametern verwenden. Die Zielsetzung dieser Lektion ist zu lernen, wie man Programme mit Parametern als Unterprogramme beim modularen Entwurf verwenden kann. Nehmen wir an, wir wollen das Programm `MUS1` so ändern,

dass die Quadratgröße frei wählbar wird. Das bedeutet, dass wir statt des Programms QUAD100 das Programm QUADRAT :GR mit dem Parameter :GR verwenden wollen. Wenn ein Unterprogramm einen Parameter hat, sollte auch sein Hauptprogramm einen Parameter besitzen, um den gewünschten Wert des Parameters beim Aufruf des Hauptprogramms angeben zu können. Somit sieht unser Programm wie folgt aus:

```
to MUS2 :GR
repeat 20 [ QUADRAT :GR fd 30 rt 10 ]
end
```

Wenn man jetzt den Befehl

```
MUS2 50
```

eintippt, wird im Register GR die Zahl 50 gespeichert und überall im Programm, wo :GR vorkommt, wird die Zahl 50 dafür eingesetzt. Somit kommt es bei der Ausführung des Programms zu Aufrufen von QUADRAT 50. Damit werden in der Schleife 20-mal Quadrate der Seitengröße 50 gezeichnet.

Aufgabe 6.2 Teste das Programm MUS2 für unterschiedliche Parametergrößen.

Wenn man jetzt auch die Entscheidung trifft, dass auch die Zahl hinter dem Befehl fd (der Parameterwert des Befehls fd) frei wählbar sein soll, kann das Programm wie folgt aussehen.

```
to MUS3 :GR :A
repeat 20 [ QUADRAT :GR fd :A rt 10 ]
end
```

Aufgabe 6.3 Erweitere MUS3 zu einem Programm MUS4, indem du auch die Anzahl der Wiederholungen des Befehls repeat frei wählbar machst.

Aufgabe 6.4 In Beispiel 5.2 haben wir das Programm RECHT :HOR :VER entwickelt, um Rechtecke mit beliebiger Seitenlänge zeichnen zu können. Ersetze in dem Programmen MUS1, MUS2 und MUS3 das Unterprogramm QUADRAT durch das Unterprogramm RECHT mit zwei Parametern. Wie viele Parameter haben dann die Hauptprogramme MUS2 und MUS3?

Beispiel 6.1 In diesem Beispiel wollen wir lernen, Blumen zu zeichnen und zwar nicht nur Blumen, die aus Kreisen bestehen. Deswegen fangen wir damit an zu lernen, wie man

Blätter zeichnen kann, die beliebig schmal sein dürfen. Ein Blatt wie in Abb. 6.1 kann man als zwei zusammengeklebte Teilkreise *A* und *B* ansehen.

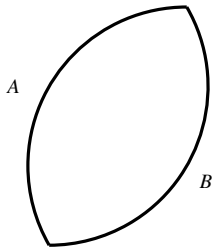

Abbildung 6.1

Einen Teilkreis kann man zum Beispiel mit folgendem Programm zeichnen:

```
repeat 120 [ fd 3 rt 1 ]
end
```

Probiere es aus.

Wir sehen, dass dieses Programm sehr ähnlich dem Programm für den Kreis ist. Anstatt 360 kleine Schritte mit jeweils 1° Drehung machen wir nur 120 kleine Schritte [`fd 3 rt 1`] und zeichnen dadurch nur ein Drittel des Kreises $\left(\frac{360}{120} = 3\right)$. Jetzt ist die Frage, wie viel man die Schildkröte drehen muss, bevor man den Teilkreis *B* für die untere Seite des Blattes zeichnet. Schauen wir uns Abb. 6.2 an.

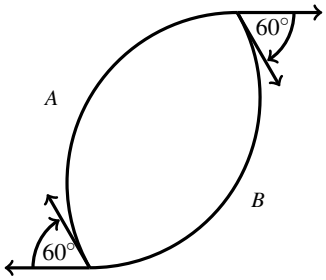

Abbildung 6.2

Wenn wir am Ende die ursprüngliche Position erreichen wollen, müssen wir die Schildkröte insgesamt, wie immer, um 360° drehen. Im Teil *A* drehen wir sie um 120° und im

Teil *B* um weitere 120°.

$$360^\circ - 120^\circ - 120^\circ = 120^\circ$$

Es bleiben also noch 120° übrig, die man gleichmäßig auf die zwei Drehungen an den Spitzen des Blattes verteilen muss.

$$\frac{120^\circ}{2} = 60^\circ$$

Damit erhalten wir folgendes Programm:

```
repeat 120 [ fd 3 rt 1 ]
rt 60
repeat 120 [ fd 3 rt 1 ]
rt 60
```

Oder noch einfacher:

```
repeat 2 [repeat 120 [ fd 3 rt 1 ] rt 60 ]
```

Probiere es aus.

Jetzt kann man sich wünschen, schmalere Blätter (die Teile A und B sind kürzer) oder breitere Blätter (die Teile A und B sind länger) zu zeichnen. Dazu kann man wieder einen Parameter verwenden. Nennen wir den Parameter zum Beispiel `:LANG` (wie Länge). Dann berechnen wir die Drehung an der Spitze des Blattes wie folgt:
Bevor man den Teil B des Blattes zeichnet, soll die Hälfte der ganzen Drehung, das heißt $\frac{360^\circ}{2} = 180^\circ$, gemacht werden. Also ist die Drehung an der Spitze des Blattes

$$180^\circ - \texttt{:LANG}.$$

Damit können wir das folgende Programm aufschreiben:

```
to BLATT1 :LANG
repeat 2 [repeat :LANG [ fd 3 rt 1 ] rt 180-:LANG ]
end
```

□

Aufgabe 6.5 Gib das Programm BLATT1 aus dem Beispiel 6.1 ein und schreibe dann:

```
BLATT1 20 BLATT1 40 BLATT1 60 BLATT1 80 BLATT1 100
```

und beobachte die Auswirkung.

Das Programm BLATT1 ermöglicht uns, die Blätter durch zwei Teilkreise eines Kreises mit Umfang $3 \cdot 360$ zu zeichnen. Das kommt dadurch, dass der zu Grunde liegende Kreis durch das Programm

```
repeat 360 [ fd 3 rt 1 ]
```

gezeichnet werden würde. Wir wollen nun die Größe des Kreises, aus dem wir Teile ausschneiden, auch frei wählbar machen. Deswegen erweitern wir BLATT1 zu dem Programm BLATT, mit dem wir beliebig große und beliebig dicke (schmale) Blätter zeichnen können.

```
to BLATT :LANG :GROSS
repeat 2 [repeat :LANG [ fd :GROSS rt 1 ] rt 180-:LANG ]
end
```

Aufgabe 6.6 Wie sehen Bilder aus, in denen man mehrfach das Programm BLATT mit einem festen Parameterwert für :LANG, aber wechselnden Parameterwerten für :GROSS, aufruft? Was ändert sich, wenn der Wert für :GROSS fest ist und nur der Wert für :LANG geändert wird?

Aufgabe 6.7 Kannst du das Programm BLATT verwenden, um Kreise beliebiger Größe zu zeichnen?

Jetzt verwenden wir das Programm BLATT als Unterprogramm zum Zeichnen von Blumen, die wie Abb. 6.3 auf der nächsten Seite aussehen.

Dabei gehen wir genauso vor, wie bei der Zeichnung der Blume mittels Kreisen.

```
to BLUMEN :ANZAHL :LANG :GROSS
repeat :ANZAHL [BLATT :LANG :GROSS rt 360/:ANZAHL ]
end
```

Damit ist jetzt BLUMEN ein Hauptprogramm mit dem Unterprogramm BLATT. Das Programm BLUMEN hat drei Parameter und das Unterprogramm BLATT hat zwei Parameter.

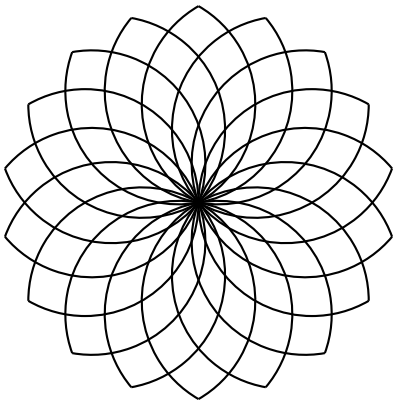

Abbildung 6.3

Jetzt wollen wir noch etwas kompliziertere zweifarbige Blumen zeichnen, indem wir das Programm BLUMEN als Unterprogramm verwenden. Die Farben sollen dabei frei wählbar sein.

```
to BLU1 :ANZAHL :LANG :GROSS :F1 :F2
setpencolor :F1
BLUMEN :ANZAHL :LANG :GROSS
rt 5
setpencolor :F2
BLUMEN :ANZAHL :LANG :GROSS
end
```

Wir beobachten hier, wie relativ der Begriff Hauptprogramm ist. Das Programm BLUMEN ist ein Hauptprogramm in Beziehung zum Unterprogramm BLATT. Andererseits ist das Programm BLUMEN ein Unterprogramm des Programms BLU1. Hier ist damit BLU1 das Hauptprogramm bezüglich der Programme BLUMEN und BLATT, obwohl BLATT nicht explizit in BLU1 aufgerufen wird, sondern unter BLUMEN versteckt bleibt. Wenn man BLU1 als ein Modul für ein noch komplexeres Programm P verwenden würde, würde BLU1 ein Unterprogramm in Bezug zum neuen Programm P werden. Alle Unterprogramme von BLU1 würden damit auch automatisch Unterprogramme von P werden.

Aufgabe 6.8 Lass dich von dem Programm BLUMEN inspirieren und zeichne eigene Muster.

Aufgabe 6.9 Erweitere das Programm BLUMEN, indem du am Ende des Programms noch einmal die Farbe änderst und das Programm MUS3 einfügst. Wie viele Parameter hat jetzt das Programm?

Aufgabe 6.10 Im Beispiel 5.3 haben wir ein Programm FELD mit drei Parametern zum Zeichnen beliebiger $X \times Y$-Felder mit wählbarer Quadratgröße :GR entwickelt. Dieses Programm FELD hat keine Unterprogramme. Deine Aufgabe ist jetzt, ein Programm mit gleicher Wirkung und den gleichen drei Parametern modular zu entwickeln. Als Basis soll das Unterprogramm QUADRAT :GR dienen. Verwende das Programm QUADRAT, um ein Programm ZEILE :GR :Y zum Zeichnen von $1 \times Y$-Felder für ein frei wählbares Y zu bauen. Verwende danach das Programm ZEILE als Baustein, um das Programm FELD modular aufzubauen.

Aufgabe 6.11 Entwickle ein Programm zum Zeichnen von $2i \times 2i$-Schachfeldern, wobei das i frei wählbar ist. Dabei soll die Farbe sowie die Größe der Quadrate (einzelne Felder) frei wählbar sein. Dein Entwurf muss modular sein. Fange mit fetten Linien beliebiger Länge an und mache mit ausgefüllten Quadraten beliebiger Größe weiter.

Hinweis für die Lehrperson Den Rest dieser Lektion empfehlen wir nur für Schüler ab dem sechsten Schuljahr. Das Ziel ist es, auf die Einführung des Konzeptes der Variable vorzubereiten. Hier geht es darum, genauer zu verstehen, wie sich bei Aufrufen von Programmen und Unterprogrammen die gespeicherten Zahlen (Inhalte) der Register ändern.

Bisher haben wir immer darauf geachtet, dass das Hauptprogramm die gleichen Parameter enthält, die seine Unterprogramme verwenden. Zum Beispiel im Programm

```
to BLUMEN :ANZAHL :LANG :GROSS
repeat :ANZAHL [ BLATT :LANG :GROSS rt 360/:LANG ]
end
```

verwenden wir das Unterprogramm BLATT mit den Parametern :LANG und :GROSS. Deswegen haben wir beide Parameter auch als Parameter des Hauptprogramms BLUMEN aufgenommen. Der Vorteil dieses Vorgehens war, dass wir anschaulich beobachten konnten, wie beim Aufruf

```
BLUMEN 12 130 2
```

den drei Parametern die Werte 12, 130 und 2, wie in Abb. 6.4 auf der nächsten Seite, zugeordnet wurden. Damit war auch sofort klar, dass die Parameter :LANG und :GROSS des Unterprogramms BLATT die Werte 130 und 2 erhielten und somit wurde bei jedem Aufruf des Unterprogramms BLATT der Befehl BLATT 130 2 ausgeführt.

Stellen wir uns jetzt vor, dass wir eine Blume zeichnen wollen, in der unterschiedliche Blätter vorkommen. Auf die Art und Weise, wie wir bisher gearbeitet haben, wäre es

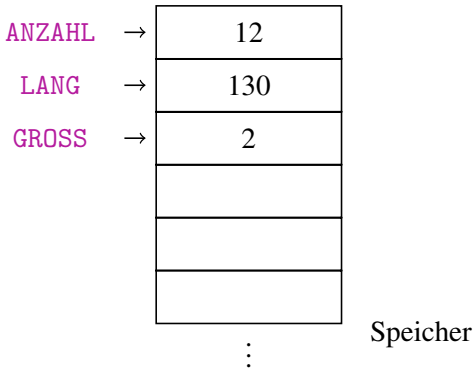

Abbildung 6.4

nicht möglich, weil wir den Parametern :LANG und :GROSS gleich am Anfang durch den Aufruf BLUMEN a b c feste Werte b und c zuordnen und dann jede Anwendung des Unterprogramms BLATT nur mit den Werten b und c als BLATT b c arbeiten muss.

Deswegen besteht die Möglichkeit, die Parameter des Hauptprogramms anders als in den Unterprogrammen zu benennen und dann mit den neuen Parameternamen die Unterprogramme aufzurufen. In unserem Beispiel kann es wie folgt aussehen.

```
to BLUMEN3 :ANZAHL :L1 :L2 :L3 :G1 :G2 :G3
repeat :ANZAHL [BLATT :L1 :G1 rt 360/:ANZAHL ]
repeat :ANZAHL [BLATT :L2 :G2 rt 360/:ANZAHL ]
repeat :ANZAHL [BLATT :L3 :G3 rt 360/:ANZAHL ]
end
```

Aufgabe 6.12 Tippe das Programm BLUMEN3 und teste es mit den Aufrufen BLUMEN3 12 100 100 100 1 2 3 und BLUMEN3 18 80 100 120 1 1.5 2.5.

Um zu verstehen, wie das Programm funktioniert, müssen wir genau beobachten, wie der Rechner bei der Ausführung des Programms mit seinen Speicherinhalten umgeht. Nach der Definition des Hauptprogramms

```
to BLUMEN3 :ANZAHL :L1 :L2 :L3 :G1 :G2 :G3
```

werden für alle sieben Parameter des Programms BLUMEN Register reserviert (Abb. 6.5(a)). Das Unterprogramm BLATT wurde schon vorher definiert und deswegen gibt es im

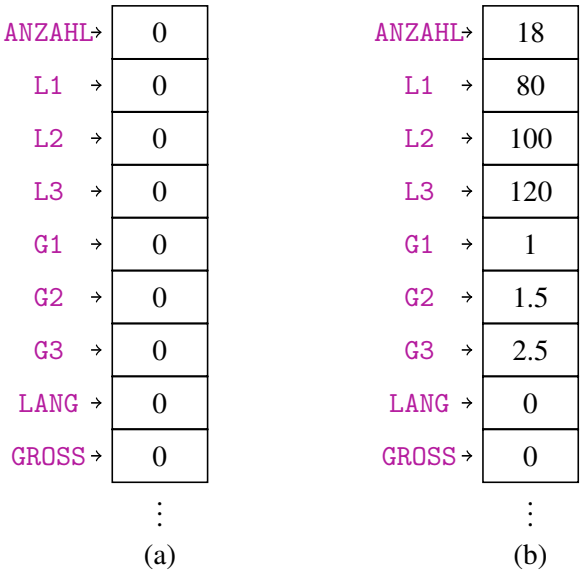

Abbildung 6.5

Speicher schon die Register mit den Namen LANG und GROSS, die den Parametern von BLATT entsprechen (Abb. 6.5(a)). Alle Register beinhalten die Zahl 0, weil die Programme zwar definiert, aber noch nicht mit konkreten Werten aufgerufen worden sind.

Nach dem Aufruf

```
BLUMEN3 18 80 100 120 1 1.5 2.5
```

erhalten die sieben Parameter des Hauptprogramms BLUMEN3 ihre Werte wie in Abb. 6.5(b) dargestellt. Die Parameternamen :LANG und :GROSS des Unterprogramms BLATT unterscheiden sich von den Parameternamen des Hauptprogramms BLUMEN3 und deswegen wird der Inhalt der entsprechenden Register LANG und GROSS nicht geändert (Abb. 6.5(b)).

Bei der Ausführung der ersten Zeile

```
repeat :ANZAHL [BLATT :L1 :G1 rt 360/:ANZAHL ]
```

des Hauptprogramms werden die Parameterwerte für ANZAHL, L1 und G1 eingesetzt und somit entsteht die Anweisung

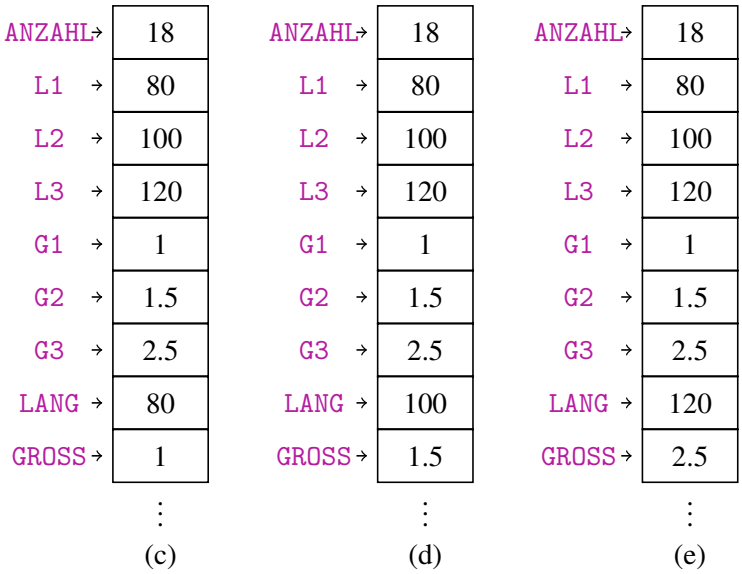

Abbildung 6.6

```
repeat 18 [BLATT 80 1 rt 360/18 ].
```

Jetzt kommt der wichtigste Punkt. Der Rechner weiß, dass der erste Parameter von `BLATT :LANG` und der zweite Parameter `:GROSS` heißt. Er ordnet nach dem Aufruf `BLATT 80 1` die Zahl `80` dem Parameter `:LANG` und die Zahl `1` dem Parameter `:GROSS` zu. Damit wird der Inhalt des Registers `LANG` auf `80` und der Inhalt des Registers `GROSS` auf `1` gesetzt (Abb. 6.6(c)). Es wird also 18-mal `BLATT 80 1` gefolgt von einer Drehung `rt 360/18` durchgeführt.

Bei der Ausführung der zweiten Zeile

```
repeat :ANZAHL [BLATT :L2 :G2 rt 360/:ANZAHL ]
```

des Hauptprogramms werden die Inhalte der Register `ANZAHL`, `L2` und `L3` (s. Abb. 6.6(c)) als entsprechende Parameterwerte eingesetzt und wir erhalten die Anweisung

```
repeat 18 [BLATT 100 1.5 360/18 ].
```

Damit wird `100` im Register `LANG` und `1.5` im Register `GROSS` gespeichert (s. Abb. 6.6(d)). Die vorherigen Inhalte `80` und `1` der Register `LANG` und `GROSS` werden damit in diesen

Registern gelöscht. Diese Werte ändern sich erst bei der Ausführung der dritten Zeile

```
repeat :ANZAHL [BLATT :L3 :G3 rt 360/:ANZAHL ].
```

Hier werden die gespeicherten Werte für `:ANZAHL`, `:L3` und `:G3` eingesetzt und damit die Anweisung

```
repeat 18 [BLATT 120 2.5 360/18 ]
```

realisiert. Dadurch wird dann `120` im Register `LANG` und `2.5` im Register `GROSS` abgelegt (s. Abb. 6.6(e))

Wir können die ganze Entwicklung des Speicherinhalts in einer Tabelle wie in Tab. 6.1 anschaulich darstellen. Die 0-te Spalte entspricht dabei der Situation nach dem Aufruf des Hauptprogramms mit konkreten Parametern. Die i-te Spalte entspricht der Speicherbelegung nach der Durchführung der i-ten Zeile des Hauptprogramms. Die Zeilen entsprechen den einzelnen Registern.

	0	1	2	3
ANZAHL	18	18	18	18
L1	80	80	80	80
L2	100	100	100	100
L3	120	120	120	120
G1	1	1	1	1
G2	1.5	1.5	1.5	1.5
G3	2.5	2.5	2.5	2.5
LANG	0	80	100	120
GROSS	0	1	1.5	2.5

Tabelle 6.1

Aufgabe 6.13 Zeichne die Entwicklung der Speicherinhalte wie in Tab. 6.1 für die Ausführung der folgenden Aufrufe des Programms `BLUMEN3`:

a) `BLUMEN3 12 100 100 100 1 2 3`

b) `BLUMEN3 15 60 80 100 2 2 2`

c) `BLUMEN3 24 90 100 110 0.5 1 2.5`

Lass den Rechner diese Aufrufe von `BLUMEN3` durchführen.

Aufgabe 6.14 Betrachte die folgenden Programme:

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

to PYR :Q1 :Q2 :Q3 :Q4
repeat 4 [ fd :Q1 rt 90 ]
repeat 4 [ fd :Q2 rt 90 ]
repeat 4 [ fd :Q3 rt 90 ]
repeat 4 [ fd :Q4 rt 90 ]
end
```

Führe den Aufruf

```
PYR 50 75 100 125
```

durch. Zeichne wie in Tab. 6.1 die Änderungen der Registerinhalte für die Parameter `:GR`, `:Q1`, `:Q2`, `:Q3` und `:Q4` ein.

Aufgabe 6.15 Schreibe ein Programm `QU4` für das Bild in Abb. 6.7. `QUADRAT :GR` soll als

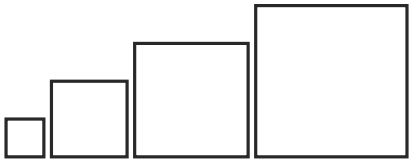

Abbildung 6.7

Unterprogramm verwenden werden und die Größe von allen vier Quadraten soll frei wählbar sein. Beschreibe dann die Entwicklung des Speicherinhalts beim Aufruf `QU4 50 100 150 200`

Aufgabe 6.16 Schreibe ein Programm `QUG` zum Zeichnen von gefüllten (z. B. schwarzen) Rechtecken mit beliebiger Größe. Nutze dieses Programm als Unterprogramm zum Zeichnen von drei gefüllten, nebeneinander liegenden Rechtecken beliebiger Größe, wie z. B. in Abb. 6.8 auf der nächsten Seite.

Abbildung 6.8

Aufgabe 6.17 Definiere ein Programm mit fünf Parametern zum Zeichnen von fünf regulären Vielecken, jeweils aus der gleichen Startposition, mit einer jeweiligen Seitengröße von 50. Die Anzahl der Ecken soll bei allen fünf wählbar sein, und die Zeichnung muss das Unterprogramm

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

fünfmal verwenden.

Alle Parameter, die im Hauptprogramm nach `to` und dem Programmnamen genannt werden, nennen wir **globale Parameter**. Im Hauptprogramm `BLUMEN3` sind `:ANZAHL`, `:L1`, `:L2`, `:L3`, `:G1`, `:G2` und `:G3` die globalen Parameter. Mit dem Aufruf des Programms `BLUMEN3` für konkrete Werte erhalten alle diese Parameter ihre Werte und werden zur Laufzeit des Programms `BLUMEN3` auch nicht mehr geändert. Wir sehen in Tab. 6.1 gut, dass sich die Inhalte der Register `ANZAHL`, `L1`, `L2`, `L3`, `G1`, `G2` und `G3` nicht ändern. Die Parameter, die durch die Unterprogramme definiert sind, nennen wir **lokale Parameter** des Hauptprogramms. Damit sind `:LANG` und `:GROSS` lokale Parameter des Hauptprogramms `BLUMEN3`. Auf der anderen Seite sind `:LANG` und `:GROSS` die globalen Parameter des Programms `BLATT`. Während der Ausführung des Unterprogramms `BLATT` ändern sich die Werte von `:LANG` und `:GROSS` nicht. Weil aber bei der Ausführung des Hauptprogramms `BLUME3` das Unterprogramm `BLATT` mehrmals mit jeweils unterschiedlichen Parametern aufgerufen werden darf, können sich die lokalen Parameter im Laufe des Hauptprogramms mehrmals ändern.

Aufgabe 6.18 Bestimme für alle in den Aufgaben 6.14, 6.15, 6.16 und 6.17 entwickelten Hauptprogramme, welche Parameter global und welche lokal bezüglich des Hauptprogramms sind.

Hinweis für die Lehrperson Am Anfang dieser Lektion haben wir die gleichen Parameternamen im Hauptprogramm wie im Unterprogramm verwendet. Damit waren diese Parameter sowohl

global als auch lokal. Was das Ganze bei der Durchführung des Programms bedeutet, werden wir erst in späteren Lektionen genauer erklären. Somit soll man in dieser Lektion beim Üben der Änderung der Register vermeiden, Programme mit gleichnamigen globalen und lokalen Variablen zu betrachten.

Wir können die lokalen Parameter geschickt benutzen, indem wir sie abwechselnd zu unterschiedlichen Zwecken verwenden. In Lektion 5 haben wir das Programm RECHT mit den Parametern :HOR und :VER definiert, das ein Rechteck der Größe $HOR \times VER$ zeichnet. Jetzt betrachten wir die Aufgabe, das Bild aus Abb. 6.9 zu zeichnen.

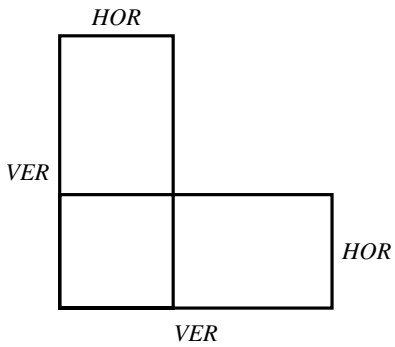

Abbildung 6.9

Das Bild kann man durch eine zweifache Anwendung von RECHT zeichnen, indem man bei der zweiten Anwendung die Rolle der Parameter vertauscht. Das Programm kann wie folgt aussehen:

```
to REC2 :A :B
RECHT :A :B
RECHT :B :A
end
```

Aufgabe 6.19 Teste das Programm für $A = 100$ und $B = 200$. Zeichne eine Tabelle (wie Tab. 6.1 auf Seite 111), die die Übergabe der Parameterwerte dokumentiert.

Aufgabe 6.20 Schreibe ein Programm zum Zeichnen rot ausgefüllter Rechtecke von frei wählbarer Größe $HOR \times VER$. Nutze dieses Programm, um Kreuze wie in Abb. 6.10 auf der nächsten Seite mit wählbarer Größe zu zeichnen.

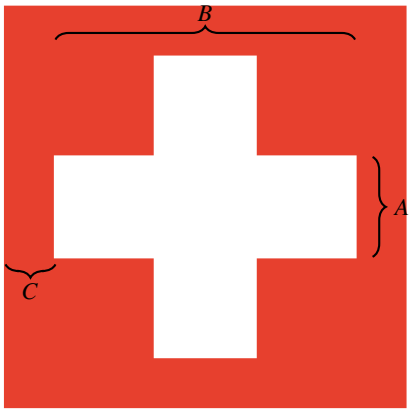

Abbildung 6.10

Zusammenfassung

Programme mit Parametern können auch Unterprogramme mit Parametern beinhalten. Dabei unterscheiden wir zwei Möglichkeiten:

1. Die Parameternamen des Unterprogramms werden auch in der Definition des Hauptprogramms verwendet. Bei einem Aufruf des Hauptprogramms mit konkreten Zahlen als Parameterwerten werden dadurch automatisch die Werte an die Unterprogramme übergeben. In diesem Fall wird kein Wert eines Parameters während der Ausführung des Hauptprogramms geändert. Dies bedeutet, dass die nach Aufruf des Hauptprogramms im Register gespeicherten Werte während der Ausführung des Hauptprogramms unverändert bleiben.

2. Einige Parameternamen von Unterprogrammen werden nicht in der Definition des Hauptprogramms verwendet. Wenn so etwas vorkommt, nennen wir die entsprechenden Parameter der Unterprogramme lokale Parameter. Die in der ersten Zeile des Hauptprogramms definierten Parameter nennen wir globale Parameter. Durch den Aufruf des Hauptprogramms mit konkreten Zahlen werden die Werte der lokalen Parameter nicht bestimmt. Erst beim Aufruf eines Unterprogramms, weist das Hauptprogramm den Parametern des Unterprogramms konkrete Werte zu. Die lokalen Parameter erhalten die Werte der globalen Parameter, die beim Aufruf an den entsprechenden Positionen der lokalen Parameter stehen. Wenn wir das

Unterprogramm mehrmals mit unterschiedlichen globalen Parametern aufrufen, ändern sich die Werte der lokalen Parameter des Hauptprogramms immer entsprechend des neuen Aufrufs. Auf diese Weise kann man ein Unterprogramm mehrmals zum Zeichnen unterschiedlich großer Objekte nutzen.

Kontrollfragen

1. Was verstehen wir unter einem modularen Entwurf?

2. Kann ein Programm gleichzeitig ein Hauptprogramm und ein Unterprogramm sein?

3. Was sind globale Parameter eines Programms und was sind lokale Parameter eines Programms? Welche Programme haben lokale Parameter?

4. Können sich die Werte der globalen Parameter während der Ausführung des Hauptprogramms ändern?

5. Wie kann es sein, dass sich die Werte der lokalen Parameter bei der Ausführung des Hauptprogramms ändern? Erkläre es an einem Beispiel.

6. Können sich die Werte der lokalen Parameter während der Ausführung des entsprechenden Unterprogramms (in dem sie definiert sind) ändern?

Kontrollaufgaben

1. Entwickle ein Programm zum Zeichnen der Bilder aus Abb. 6.11(a) und Abb. 6.11(b). Die Bilder sollen eine frei wählbare Größe haben.

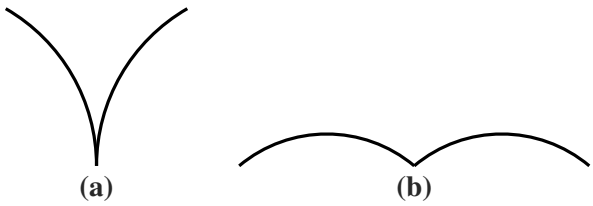

Abbildung 6.11

2. Zeichne das Bild aus Abb. 6.12.

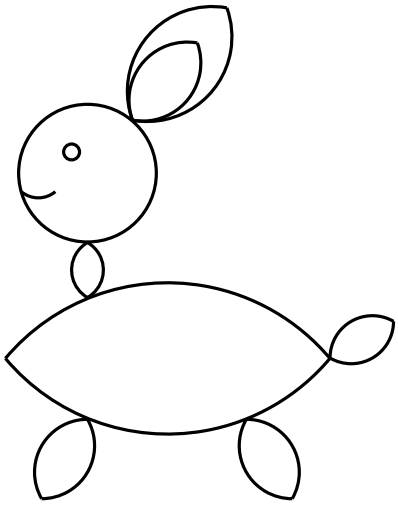

Abbildung 6.12

3. Entwickle in modularer Art ein Programm, das Bilder wie in Abb. 3.8 auf Seite 69 zeichnen kann. Dabei sollen die Größe der schwarzen Quadrate sowie ihre Anzahl (die Höhe des Bildes) frei wählbar sein.

4. Entwickle ein Programm zum Zeichnen roter Kreuze der Form wie in Abb. 3.11 auf Seite 70. Dabei sollen die Größe sowie die Anzahl der Kreuze frei wählbar sein.

5. Das Programm

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

zeichnet Kreise in beliebiger Farbe und beliebiger Größe. Entwickle ein Hauptprogramm KETTE, das KREIS2 als Unterprogramm verwendet. Dabei sollte das Programm KETTE eine Folge von vier Kreisen wie in Abb. 6.13 auf der nächsten Seite zeichnen. Die Größe und die Farbe jedes einzelnen Kreises sollen auch frei wählbar sein. Rufe das Programm mit ausgewählten Parametern auf und stelle die Entwicklung der Registerinhalte wie in Tab. 6.1 dar.

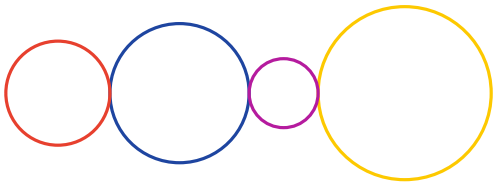

Abbildung 6.13

6. Entwickle ein Programm zum Zeichnen des Bildes in Abb. 4.8 auf Seite 82. Die Farbe des Bildes und die Seitengröße der 6-Ecke sollen frei wählbar sein. Als Unterprogramm zum Zeichnen einzelner 6-Ecke soll folgendes Programm verwendet werden.

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

7. In Abb. 5.1 auf Seite 86 sind fünf Quadrate unterschiedlicher Größe vom gleichen Startpunkt (die Ecke links unten) gezeichnet. Entwickle modular ein Programm, das fünf solcher Quadrate beliebiger Größe und Farbe zeichnet. Dabei soll das Hauptprogramm das Programm `QUADRAT :GR` als Unterprogramm verwenden.

   Ruf dein Programm auf, um Quadrate der Größe 50, 70, 100, 140 und 190 zu zeichnen und stell die Entwicklung der Registerinhalte wie in Tab. 6.1 dar.

8. Entwickle ein Programm, ähnlich zu dem Programm aus Kontrollaufgabe 7, mit dem du statt Quadraten regelmäßige 8-Ecke zeichnest.

Lösungen zu ausgesuchten Aufgaben

Aufgabe 6.7
Wenn man für den Parameter `:LANG` den Wert 180° wählt, erhält man einen Kreis. Der Parameter `:GROSS` bleibt frei wählbar und bestimmt die Größe des Kreises. Somit zeichnen beide Programme

`BLATT 180 x` und `KREISE x`

das gleiche Bild und zwar einen Kreis als regelmäßiges 360-Eck mit der Seitenlänge `x`.

Aufgabe 6.15
Weil wir alle vier Größen frei wählbar haben wollen, verwenden wir vier Parameter, für jeden Aufruf des Unterprogramms QUADRAT einen anderen. Dabei achten wir noch darauf, dass zwischen den vier Quadraten kleine Abstände entstehen.

```
to QU4 :G1 :G2 :G3 :G4
QUADRAT :G1
rt 90 pu fd :G1+3 pd lt 90
QUADRAT :G2
rt 90 pu fd :G2+3 pd lt 90
QUADRAT :G3
rt 90 pu fd :G3+3 pd lt 90
QUADRAT :G4
end
```

Für die sieben Zeilen des Programms QU4 erhalten wir beim Aufruf QU4 50 100 150 200 die Entwicklung der Registerinhalte G1, G2, G3 und G4 wie in Tab. 6.2 dargestellt.

	0	1	2	3	4	5	6	7
G1	50							
G2	100							
G3	150							
G4	200							
GR	0	50		100		150		200

Tabelle 6.2

Um es anschaulicher zu machen, schreiben wir die Werte nur dann in die Tabelle, wenn sie sich geändert haben. Ansonsten wären die Werte in den ersten vier Zeilen der Tabelle alle gleich.

Aufgabe 6.17
Weil man sich nach dem Zeichnen eines Vielecks mittels des Programms VIELECK immer in der ursprünglichen Startposition befindet, ist diese Aufgabe sehr leicht lösbar.

```
to VE5 :E1 :E2 :E3 :E4 :E5
VIELECK :E1 VIELECK :E2 VIELECK :E3
VIELECK :E4 VIELECK :E5
end
```

Damit sind :E1, :E2, :E3, :E4 und :E5 globale Parameter des Programms VE5 und :ECKE (aus dem Programm VIELECK) ist der lokale Parameter des Programms VE5.