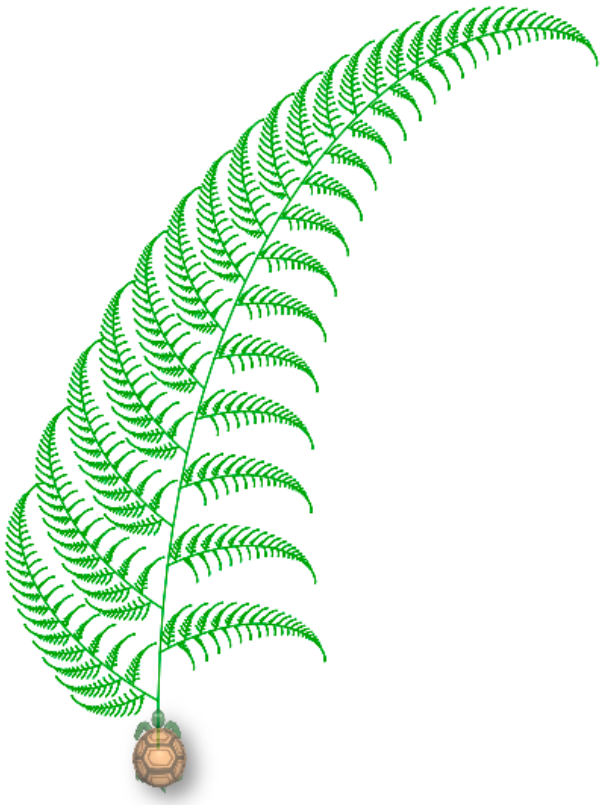
Rekursion – Teil 1

Mit Logo lassen sich unglaublich leicht die tollsten Bilder erzeugen. Du hast bereits Wiederholungsanweisungen benutzt, um schöne Figuren zu zeichnen. Es geht aber noch besser! Zum Beispiel ist folgenden Farnblatt ganz leicht mit Logo zu programmieren:

Viele komplizierte Formen wie dieses Farnblatt bestehen aus einfachen Formen desselben Typs, auf die sich die komplizierte Form zurückführen lässt.

Aufgabe 1

Schau dir das Farnblatt einmal genau an! Was ist hier die einfache Grundform, aus der jedes Farnblatt aufgebaut ist?



Wir wollen nun versuchen, das Farnblatt zu zeichnen, indem wir die komplizierte Form durch Rückführung auf die einfache Grundform beschreiben.

Die Grundfigur soll aus einem einzigen Farnstengel bestehen, der durch eine leicht gekrümmte Linie dargestellt wird:

```
lb
setze "seite 100

vw :seite
re 2
vw :seite
rw :seite
li 2
rw :seite
```

Dazu bauen wir unsere Prozedur "farn" wie folgt auf:

```
lerne farn :x :o
    wenn (:o<=0)
    [
        #Abbruch
    ]
    [
        vw :x
        li 60
        # tue dasselbe nocheinmal, aber kleiner
        farn (:x*0.3) (:o-1)
        re 60

        re 2
        vw :x

        re 60
        # tue dasselbe nocheinmal, aber kleiner
        farn (:x*0.2) (:o-1)
        li 60

        re 2
        # tue dasselbe nocheinmal, aber kleiner
        farn (:x*0.9) (:o-1)
        li 2

        sh
        rw :x
        li 2
        rw :x
        sa
    ]
Ende
```

Die Grundfigur wird also an drei Stellen unterbrochen, den sogenannten rekursiven Aufrufpunkten ("Selbstaufrufpunkten"): Zeilen 9/10, 17/18 und 22/23.

An diesen Stellen erfolgt ein Selbstaufruf, also ein Aufruf der Prozedur `farn` durch die Prozedur `farn`. Anschaulich ist das am Besten mit einem Echo vergleichbar.

Aber wie ein Echo verhallt, so soll auch der Selbstaufruf nicht unendlich fortgesetzt werden. Dazu bedarf es einer Abbruchbedingung, die hier in den Zeilen 2-5 programmiert ist: Nur wenn der als Ordnung übergebene Parameter `:o` größer als null ist, kommt es zum Abarbeiten der Zeilen 7-30, ansonsten erfolgt der Abbruch.

Zur Formulierung der Abbruchbedingung wird der `wenn`-Befehl verwandt:

```
wenn (<Prüfbedingung>)
[
    <Dann-Befehlsliste>
]
[
    <Sonst-Befehlsliste>
]
```

Wenn die Prüfbedingung den Wert `wahr` ergibt, werden die Dann-Befehle ausgeführt. Wenn die Prüfbedingung den Wert `falsch` ergibt, werden die Sonst-Befehle ausgeführt. Die Angabe der Sonst-Befehlsliste ist optional.

Aufgabe 2

Öffne die Datei "Prozeduren xLOGO 05.lgo" und gib den folgenden Befehl ein:

```
lb lt verfolge farn 100 1
```

Betrachte das Ergebnis. Notiere die Textausgabe und versuche zu verstehen, was hier passiert ist.

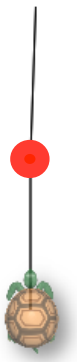
```
farn 100 1
  farn 30 0
  farn 20 0
  farn 90 0
```

Erhöhe nun den Wert des Parameters `:o` auf 2. Betrachte wieder das Ergebnis, notiere die Textausgabe und versuche zu verstehen, was hier passiert ist.

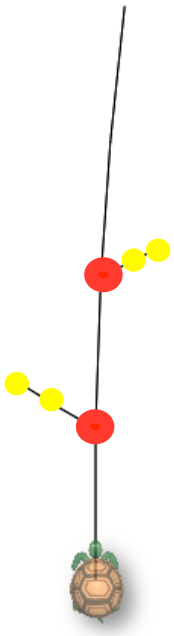
```
farn 100 1
  farn 30 0
  farn 20 0
  farn 90 0
farn 100 2
farn 100 2
  farn 30 1
    farn 9 0
    farn 6 0
    farn 27 0
  farn 20 1
    farn 6 0
    farn 4 0
    farn 18 0
  farn 90 1
    farn 27 0
    farn 18 0
    farn 81 0
```

Folgende Abbildungen zeigen die jeweils erzeugte Grafik:

`verfolge farn 100 1`



`verfolge farn 100 2`



Markiere jeweils in den Abbildungen die rekursiven Aufrufpunkte in den Abbildungen.

Spielanleitung: "Turtle-Spiel"

Rekursion ist nicht immer sofort verständlich. Daher wollen wir uns die Zeit nehmen, und selbst einmal die Turtle spielen. Du benötigst:

1 grüne Karte
3 gelbe Karten
9 rote Karten

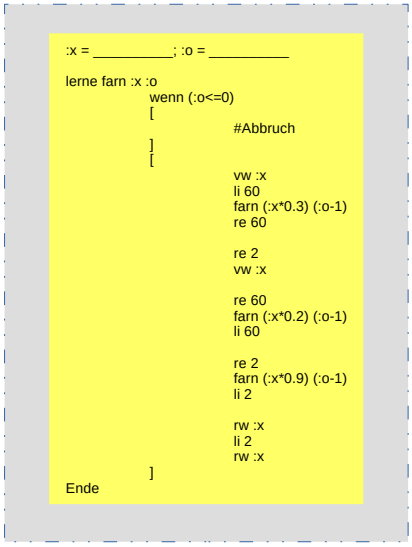
Wir simulieren nun den Prozedurbefehl: `farn 100 2`

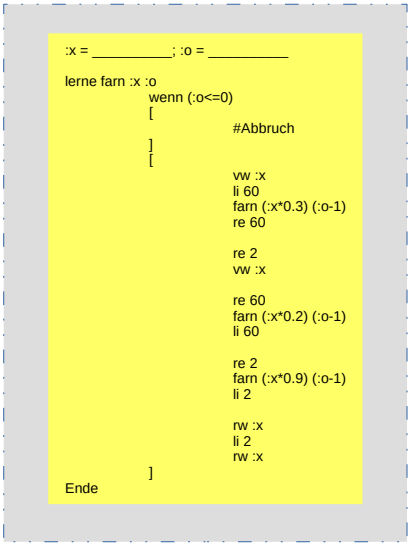
Gehe dazu wie folgt vor:

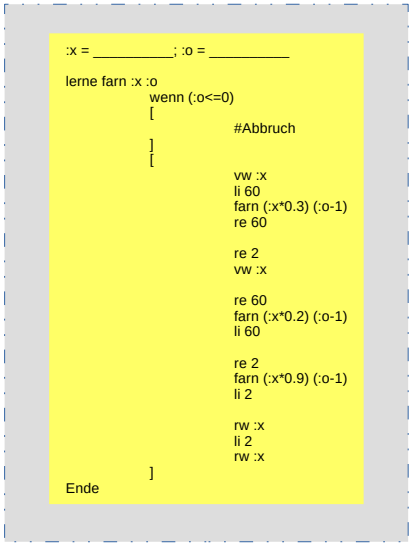
(1) Beginne nun mit der grünen Karte, wobei zu Anfang `:x = 100` und `:o = 2` gilt.
(2) Schreibe alle Turtle-Bewegungen untereinander! Abgearbeitete Zeilen werden auf der Karte abgehakt.
(3) Wenn der rekursive Aufruf der "`farn`"-Methode kommt, nimm dir eine neue Karte und lege diese zunächst auf die alte Karte. Dabei gilt:
    (a) auf eine grüne Karte (Ordnung = 2) folgt eine gelbe Karte (Ordnung = 1)
    (b) auf eine gelbe Karte (Ordnung = 1) folgt eine rote Karte (Ordnung = 0)
(4) Gehe dann vor wie unter (2).
(5) Wenn eine Karte vollständig abgearbeitet ist, dann nimm sie vom Stapel und lege sie zur Seite. Setze dann dein vorgehen mit der darunterliegenden Karte fort.
(6) Du bist fertig, wenn alle Karten fertig abgearbeitet sind und der Stapel leer ist.

Spielkarten:

```
:x = __________; :o = __________

lerne farn :x :o
        wenn (:o<=0)
        [
                #Abbruch
        ]
        [
                vw :x
                li 60
                farn (:x*0.3) (:o-1)
                re 60

                re 2
                vw :x

                re 60
                farn (:x*0.2) (:o-1)
                li 60

                re 2
                farn (:x*0.9) (:o-1)
                li 2

                rw :x
                li 2
                rw :x
        ]
Ende
```







```
:x = __________; :o = __________

lerne farn :x :o
        wenn (:o<=0)
        [
                #Abbruch
        ]
        [
                vw :x
                li 60
                farn (:x*0.3) (:o-1)
                re 60

                re 2
                vw :x

                re 60
                farn (:x*0.2) (:o-1)
                li 60

                re 2
                farn (:x*0.9) (:o-1)
                li 2

                rw :x
                li 2
                rw :x
        ]
Ende
```

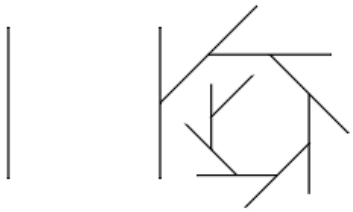
Liste der Turtle-Bewegungen:

```
vw 100
li 60
    vw 30
    li 60
        # Abbruchbedingung erfüllt -> Rückkehr
    re 60
    re 2
    vw 30
    re 60
        # Abbruchbedingung erfüllt -> Rückkehr
    li 60
    re 2
        # Abbruchbedingung erfüllt -> Rückkehr
    li 2
    rw 30
    li 2
    rw 30 # Prozedur abgearbeitet
re 60
re 2
vw 100
re 60
    vw 20
    li 60
        # Abbruchbedingung erfüllt -> Rückkehr
    re 60
    re 2
    vw 20
    re 60
        # Abbruchbedingung erfüllt -> Rückkehr
    li 60
    re 2
        # Abbruchbedingung erfüllt -> Rückkehr
    li 2
    rw 20
    li 2
    rw 20 # Prozedur abgearbeitet
li 60
re 2
    vw 90
    li 60
        # Abbruchbedingung erfüllt -> Rückkehr
    re 60
```

```
    re 2
    vw 90
    re 60
        # Abbruchbedingung erfüllt -> Rückkehr
    li 60
    re 2
        # Abbruchbedingung erfüllt -> Rückkehr
    li 2
    rw 90
    li 2
    rw 90 # Prozedur abgearbeitet
li 2
rw 100
li 2
rw 100 # Prozedur abgearbeitet
```

Anleitung für die Programmierung rekursiver Prozeduren

(1) Überlege dir, aus welcher Grundfigur die Gesamtfigur aufgebaut ist.
Bei folgender Figur "`linienMuster`" ist die Grundfigur offensichtlich eine Linie:



(2) Schreibe zunächst eine Prozedur, welche dir diese Grundfigur zeichnet. Die Turtle soll am Ende wieder an der Ausgangsposition stehen.
Im gegebenen Beispiel sähe die Grundfigur so aus:

```
vw :seite
rw :seite
```

(3) Überlege dir, an welchen Stellen du die rekursiven Aufrufpunkte brauchst.
Im gegebenen Beispiel wäre der einzige rekursive Aufrufpunkt genau in der Mitte.

(4) Zerlege die Grundfigur so, dass die Turtle an diesen rekursiven Aufrufpunkten immer einen Zwischenstopp einlegt.
Im gegebenen Beispiel sähe das dann so aus:

```
vw :seite/2
# rekursiver Aufrufpunkt
vw :seite/2
rw :seite
```

(5) Füge die rekursiven Aufrufpunkte hinzu. Die Grundfigur darf dadurch nicht beeinflusst werden.
Im gegebene Beispiel sähe das dann so aus:

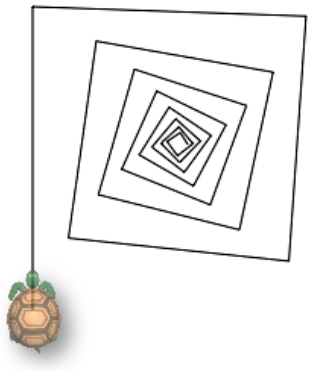
```
vw :seite/2
re 45 # würde Grundfigur verändern ...
linienMuster :seite*0.9 :ordnung-1 # rekursiver Aufruf
li 45 # ... verhindert Veränderung der Grundfigur
vw :seite/2
rw :seite
```

(6) Schließlich muss noch die Abbruchbedingung ergänzt werden.
Die vollständige Prozedur "`linienMuster`" sieht dann wie folgt aus:

```
lerne linienMuster :seite :ordnung
   wenn (:ordnung > 0)
   [
       vw :seite/2

       re 45 # würde Grundfigur verändern ...
       linienMuster :seite*0.9 :ordnung-1 # rekursiver Aufruf
       li 45 # ... verhindert Veränderung der Grundfigur

       vw :seite/2
       rw :seite
]
ENDE
```

Aufgabe 3

Erzeuge folgende Figur mithilfe einer rekursiven Prozedur! Notiere die zugehörige Prozedur.

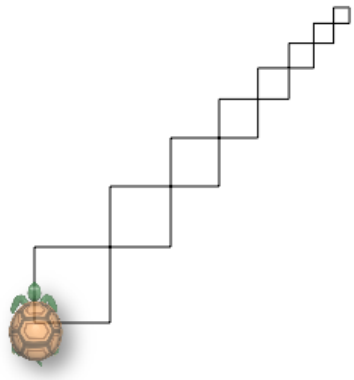


Lösung:

```
lerne tunnel :l :o [ :k 91  ]
wenn (:o >= 0)
 [
   vw :l
   re :k
   (tunnel (:l*0.9) (:o-1) :k)
   li :k
   rw :l
 ]
Ende
```

Aufgabe 4

Erzeuge folgende Figur mithilfe einer rekursiven Prozedur! Notiere die zugehörige Prozedur.

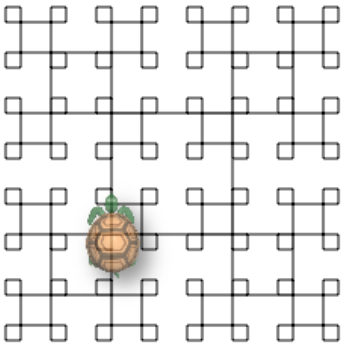


Lösung:

```
lerne quadrate :x :o
wenn (:o >= 0)
 [
  vw :x
  re 90
  vw :x

  li 90
  quadrate (:x*0.8) (:o-1)
  re 90

  re 90
  vw :x
  re 90
  vw :x
  re 90
]
Ende
```

Aufgabe 5

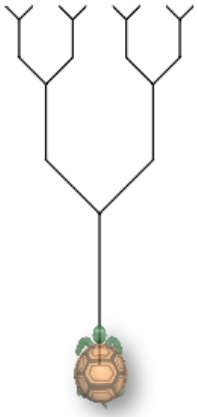
Erzeuge folgende Figur mithilfe einer rekursiven Prozedur! Notiere die zugehörige Prozedur.



Lösung:

Aufgabe 6

Erzeuge folgende Figur mithilfe einer rekursiven Prozedur! Notiere die zugehörige Prozedur.



Lösung:

```
lerne baum :x :o
wenn (:o >= 0)
 [
   vw :x
   li 45
   vw (:x/3)

   re 45
   baum (:x*0.5) (:o-1)
   li 45

   rw (:x/3)
   re 90
   vw (:x/3)

   li 45
   baum (:x*0.5) (:o-1)
   re 45

   rw (:x/3)
   li 45
   rw :x
 ]
Ende
```

Aufgabe 7

Experimentiere selbst! Schreibe eigene rekursive Figuren und schau dir an, was dabei herauskommt. Schau dir auch die Figuren deiner Sitznachbarn an - dazu darfst du auch aufstehen!