

# Struktogramme zur Modulsammlung POW

## Sequenzen

Modul	Struktogramm
Dreieck.mod	<div><div><b>Dreieck</b></div><div><div>Eingabe: Seitenlängen a, b, c</div><div>u := a + b + c</div><div>s := u / 2</div><div>F := Wurzel( s*(s-a)*(s-b)*(s-c)) (Heronische Formel)</div><div>alpha:= arcsin(2*F/(b*c)) * 180 / Pi</div><div>beta := arcsin(2*F/(a*c)) * 180 / Pi</div><div>gamma := arcsin(2*F/(a*b)) * 180 / Pi</div><div>Ausgabe: Umfang</div><div>Ausgabe: Fläche</div><div>Ausgabe: Winkel alpha, beta, gamma</div></div></div>
Kugel.mod	<div><div><b>Kugel</b></div><div><div>Eingabe: Radius r</div><div>d := 2*r</div><div>V := (4*3.1415*r*r*r) / 3</div><div>O := 4*3.1415*r*r</div><div>Ausgabe: Durchmesser d</div><div>Ausgabe: Oberfläche O</div><div>Ausgabe: Volumen V</div></div></div>

Mwst.mod

### Mehrwertsteuer

Eingabe: Nettobetrag netto

steuer := netto \* 0.16

brutto := netto + steuer

Ausgabe: netto

Ausgabe: steuer

Ausgabe: brutto

Spar.mod

### Sparschwein

Eingabe: Anzahl der Pfennige pf1

Eingabe: Anzahl der 2-Pf.-Stücke pf2

Eingabe: Anzahl der 5-Pf.-Stücke pf5

Eingabe: Anzahl der 10-Pf.-Stücke pf10

Eingabe analog für pf20, pf 50, E1

Eingabe: Anzahl der 2-Euro-Stücke E2

Summe := pf1+2\*pf2+5\*pf5+...+200\*E2

Summe := Summe / 100

Ausgabe: Summe in €

# ALTERNATIVE

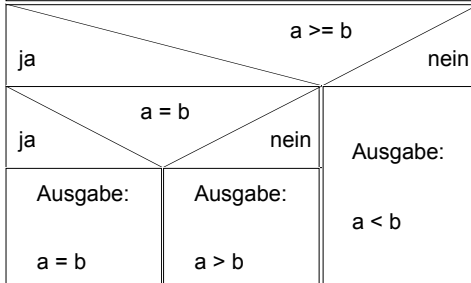
Modul	Struktogramm
Quadratg.mod	<div><div><div>Quadratische Gleichung</div><div>Ausgabe: Quadratische Gleichungen <math>x^2 + px + q = 0</math></div><div>Eingabe: p</div><div>Eingabe: q</div><div><math>D := (p/2)^2 - q</math></div><div><div><div><div><div><div><math>D &lt; 0</math></div><div>ja</div><div>nein</div></div><div><div>Ausgabe: Es gibt für diese Gleichung keine Lösung.</div><div><div><div><div><math>D = 0</math></div><div>ja</div><div>nein</div></div><div><div><div><div><math>x := -(p/2)</math></div><div><div>Ausgabe: Lösung x</div></div></div><div><div><div><math>x1 := -(p/2) + \text{Wurzel}(D)</math></div><div><math>x2 := -(p/2) - \text{Wurzel}(D)</math></div><div>Ausgabe: Lösungen x1 und x2</div></div></div></div></div></div></div></div></div></div></div></div></div></div>
Wurzel.mod	<div><div><div>Wurzel</div><div>Eingabe: Wurzelradikant z</div><div><div><div><div><math>z \geq 0</math></div><div>ja</div><div>nein</div></div><div><div><div><math>w := \text{Wurzel}(z)</math></div><div>Ausgabe: w</div></div><div><div>Ausgabe: "Wurzel nicht definiert."</div></div></div></div></div></div></div>

Vergl.mod

### Größenvergleich

Eingabe: Zahl a

Eingabe: Zahl b



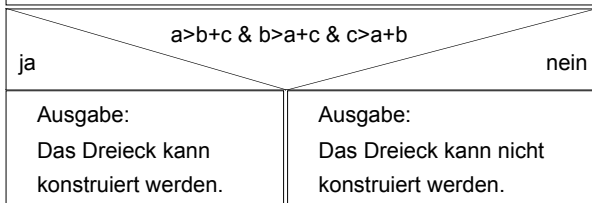
Verzweig.mod

### Dreiecksungleichung

Eingabe:  
Seitenlänge a

Eingabe:  
Seitenlänge b

Eingabe:  
Seitenlänge c



## SELEKTION

Modul	Struktogramm																		
Kino.mod	<div> <div>Kinokartenpreis</div> <div> <div>Kinopreis Grundpreis = 10 Euro</div> <div> <div>Ausgabe 1..Kurzfilm 2..Normal 3..Überlänge</div> <div>Eingabe: Wahl</div> <table> <tr> <td>Fall 1</td><td>Fall 2</td><td>Fall 3</td><td>Wahl sonst</td></tr> <tr> <td>p:=p*0.5</td><td>p:=p*1</td><td>p:=p*1.5</td><td>Fehler p:=p*2</td></tr> </table> <div> <div>Ausgabe 1..Kinder bis 10 Jahre 2..Kinder 10 bis 14 Jahre 3..Studenten / Rentner 4..Erwachsene</div> <div>Eingabe: Wahl</div> <table> <tr> <td>Fall 1</td><td>Fall 2</td><td>Fall 3</td><td>Fall 4</td><td>Wahl sonst</td></tr> <tr> <td>p:=p*0.5</td><td>p:=p*0.75</td><td>p:=p*0.9</td><td>p:=p*1.1</td><td>Fehler p:=p*2</td></tr> </table> <div>Ausgabe Preis p</div> <div>Ende</div> </div> </div> </div> </div>	Fall 1	Fall 2	Fall 3	Wahl sonst	p:=p*0.5	p:=p*1	p:=p*1.5	Fehler p:=p*2	Fall 1	Fall 2	Fall 3	Fall 4	Wahl sonst	p:=p*0.5	p:=p*0.75	p:=p*0.9	p:=p*1.1	Fehler p:=p*2
Fall 1	Fall 2	Fall 3	Wahl sonst																
p:=p*0.5	p:=p*1	p:=p*1.5	Fehler p:=p*2																
Fall 1	Fall 2	Fall 3	Fall 4	Wahl sonst															
p:=p*0.5	p:=p*0.75	p:=p*0.9	p:=p*1.1	Fehler p:=p*2															
Noten.mod	<div> <div>Noten</div> <div> <div>Eingabe: notenpunkte</div> <table> <tr> <td>Fall 1</td><td>Fall 2</td><td>Fall 3</td><td>Fall 4</td><td>Fall 5</td><td>Fall 6</td><td>notenpunkte sonst</td></tr> <tr> <td>Aus- gabe: sehr gut</td><td>Aus- gabe: gut</td><td>Aus- gabe: be- friedigend</td><td>Aus- gabe: aus- reichend</td><td>Aus- gabe: mangel- haft</td><td>Aus- gabe: unge- nügend</td><td>Aus- gabe: Fehler</td></tr> </table> </div> </div>	Fall 1	Fall 2	Fall 3	Fall 4	Fall 5	Fall 6	notenpunkte sonst	Aus- gabe: sehr gut	Aus- gabe: gut	Aus- gabe: be- friedigend	Aus- gabe: aus- reichend	Aus- gabe: mangel- haft	Aus- gabe: unge- nügend	Aus- gabe: Fehler				
Fall 1	Fall 2	Fall 3	Fall 4	Fall 5	Fall 6	notenpunkte sonst													
Aus- gabe: sehr gut	Aus- gabe: gut	Aus- gabe: be- friedigend	Aus- gabe: aus- reichend	Aus- gabe: mangel- haft	Aus- gabe: unge- nügend	Aus- gabe: Fehler													

Taste.mod

Tastenerkennung			
Eingabe: Taste			
Taste			
'A' .. 'Z'	'a' .. 'z'	'!' .. '0'	sonst
Ausgabe: Großbuchstabe	Ausgabe: Kleinbuchstabe	Ausgabe: Ziffer	Ausgabe: anderes Zeichen

## WIEDERHOLANWEISUNGEN

Modul	Struktogramm
Abbr1.mod	<div><div><b>WDH1</b></div><div><div>I := 10</div><div><div>Ausgabe: I , I≤</div><div>I := I - 2</div></div><div>Wdh. bis I &lt; 0</div></div></div>
Abbr2.mod	<div><div><b>WDH2</b></div><div><div>I := 10</div><div><div>Solange I &gt; 0 tue</div><div><div>Ausgabe: I, Wurzel( I )</div><div>I := I - 3</div></div></div></div></div>
Anlage.mod	<div><div><b>Anlage</b></div><div><div>Guthaben := 0</div><div>Jahr := 0</div><div><div>Solange Guthaben &lt; 20.000 tue</div><div><div>Guthaben := (2000 + Guthaben) * 1.04</div><div>Jahr := Jahr + 1</div><div>Ausgabe: Jahr</div><div>Ausgabe: Guthaben</div></div></div></div></div>

## Euler.mod

### Eulersche Zahl

$e := 1$

$\text{nenner} := 1$

$n := 1$

$\text{hilf} := e$

$e := e + 1/\text{nenner}$

$n := n + 1$

$\text{nenner} := \text{nenner} * n$

$\text{differenz} := |\text{hilf} - e|$

Wdh.

bis  $\text{differenz} < 0.001$

Ausgabe:

Nach  $n$  Durchläufen wurde  
 $e$  berechnet.

## Heron.mod

### Heron-Verfahren

Ausgabe: Heron-Verfahren zur Wurzelberechnung

Eingabe: Radikant  $r$   
( $r \geq 0$ )

$\text{ers} := r / 2$

$\text{zrs} := r / \text{ers}$

Solange  $|r - \text{zrs}| > 0.0001$  tue

$\text{ers} := (\text{ers} + \text{zrs}) / 2$

$\text{zrs} := r / \text{ers}$

Ausgabe: Ergebnis  $\text{zrs}$

## Krypto.mod

### Kryptogramme

Ausgabe: "KRYPTOGRAMM"

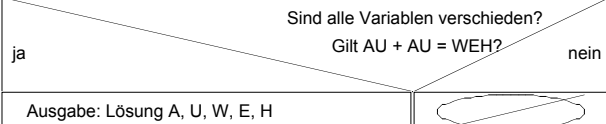
W := 1

Für A von 5 bis 9 tue

Für U von 0 bis 9 tue

Für E von 0 bis 9 tue

Für H von 0 bis 9 tue



Ausgabe: Lösung A, U, W, E, H

## Wechsel.mod

### Wechselgeld

Eingabe: Betrag

rest := betrag

Solange rest  $\geq$  200 tue

Ausgabe: 2€ - Münze

rest := rest - 200

solange rest  $\geq$  100 tue

Ausgabe 1€ - Münze

rest := rest - 100

...

Solange rest  $\geq$  2 tue

Ausgabe: 2-Cent-Münze

rest := rest - 2

solange rest  $\geq$  1 tue

Ausgabe: 1-Cent-Münze

rest := rest - 1



# DATENTYPEN

Modul	Struktogramm
Bruch.mod	<div data-bbox="491 412 1005 1169"> <p><b>Bruch-Record</b></p> <pre> graph TD     A[Eingabe: b.zaehler] --&gt; B[Eingabe: b.nenner]     B --&gt; C[t := 1]     C --&gt; D{b.zaehler &gt; b.nenner}     D -- ja --&gt; E[h := b.nenner DIV 2]     D -- nein --&gt; F[h := b.zaehler DIV 2]     E --&gt; G[Für i := 1 bis h tue]     F --&gt; G     G --&gt; H{(b.nenner MOD i = 0) &amp; (b.zaehler MOD i = 0)}     H -- ja --&gt; I[t := i]     H -- nein --&gt; J(( ))     I --&gt; K[b.zaehler := b.zaehler DIV t]     J --&gt; K     K --&gt; L[b.nenner := b.nenner DIV t]     L --&gt; M[Ausgabe: b.zaehler]     M --&gt; N[Ausgabe: b.nenner]         </pre> </div>
Codemono1.mod	<div data-bbox="491 1258 791 1518"> <p><b>Codierung ProgMain</b></p> <pre> graph TD     A[Eingabe: Otext] --&gt; B[l := Länge(Otext)]     B --&gt; C[Codiere]     C --&gt; D[Ausgabe: Ctext]         </pre> </div> <div data-bbox="491 1563 979 1944"> <p><b>Codierung Prozedur Codiere</b></p> <pre> graph TD     A[Für i := 0 bis l-1 tue] --&gt; B[h := Ordnungszahl(Otext[i]) + 3 (*Ersetzt man den Wert 3 durch einen anderen, wird der Schlüssel geändert*)]     B --&gt; C{h &gt; 122}     C -- ja --&gt; D[h := h - 26]     C -- nein --&gt; E(( ))     D --&gt; F[Ctext[i] := ASCII-Zeichen mit der Nummer h]     E --&gt; F         </pre> </div>

## Cramer.mod

### Cramer

Ausgabe:

"Lösen von linearen Gleichungssystemen 2.Grades"

Ausgabe:

"Geben sie bitte die Koeffizienten ein!"

Für z := 0 bis 1 tue

Für s := 0 bis 2 tue

Eingabe: GLS[s,z]

$D := (GLS[0,0]*GLS[1,1] - GLS[0,1]*GLS[1,0])$

$D1 := (GLS[2,0]*GLS[1,1] - GLS[2,1]*GLS[1,0])$

$D2 := (GLS[0,0]*GLS[2,1] - GLS[0,1]*GLS[2,0])$

D = 0

ja

nein

Ausgabe:  
"Das GLS ist  
nicht lösbar"

ja

nein

$D1 = 0 \ \& \ D2 = 0$

Ausgabe:  
"Das GLS ist  
mehrdeutig  
lösbar!"

$x := D1 / D$

$y := D2 / D$

Ausgabe: Lösung x

Ausgabe: Lösung y

## Datum.mod

### Datum

ja

nein

Eingabe: dat.tag  
dat.monat  
dat.jahr

$c := \text{dat.jahr} \text{ DIV } 100$

$j := \text{dat.jahr} \text{ DIV } 100$

j = 0

ja

nein

$c \text{ MOD } 4 = 0$

ja

nein

$j \text{ MOD } 4 = 0$

ja

nein

schaltjahr := TRUE

schaltjahr := FALSE

schaltjahr := TRUE

schaltjahr := FALSE

Falls dat.monat

Fall 1

Fall 2

Fall 3

Fall 4

schaltjahr

schaltjahr

3: k := 2  
4: k := 5  
5: k := 0  
6: k := 3  
7: k := 5

8: k := 1  
9: k := 4  
10: k := 6  
11: k := 2  
12: k := 4

ja

ja

nein

nein

k := 5

k := 6

k := 1

k := 2



## Laufschrift.mod

### Laufschrift

Eingabe: Zeichenkette text

laenge := Länge(text)

Für i := 1 bis (60-laenge) tue

Füge ein Leerzeichen  
an die 1.Stelle von text ein

Ausgabe: text  
(\*an eine feste Stelle zum Überschreiben\*)

## Maggi.mod

### Magisches Quadrat

Ausgabe: Magische Quadrate

Für z:=0 bis 2 tue

Für s:=0 bis 2 tue

Eingabe: mq[s,z]

Für z:=0 bis 2 tue

Für s:=0 bis 2 tue

Ausgabe: mq[s,z]

Für i:=0 bis 2 tue

summe[0]:= summe[0]+mq[0,i]

...

summe[7]:= summe[7]+mq[2-i,i]

summe[0]=summe[1] &  
... & summe[6]=summe[7]

ja

nein

Ausgabe:  
Es ist ein  
magisches Quadrat.

Ausgabe:  
Es ist kein  
magisches Quadrat.

## Pascal.mod

### Pascalsches Dreieck

Eingabe: Anzahl der Zeilen (<12) zeilen

Für n := 0 bis zeilen tue

dreieck[0,n] := 1

dreieck[n,n] := 1

Für n := 2 bis zeilen tue

Für k := 1 bis n-1 tue

dreieck[n,k] := dreieck[n-1,k-1] + dreieck[n-1,k]

Für n := 0 bis zeilen tue

Für k := 1 bis n tue

Ausgabe: dreieck[n,k]

Schreibmaschine.  
mod

### Schreibmaschine

Eingabe: Zeichenkette text

laenge := Länge(text)

Für i := 1 bis laenge tue

Kopiere das i. Zeichen von text nach letter

Ausgabe: letter

Runters.mod

### Reihenuntersuchung

Für i := 0 bis 149 tue

Eingabe: Messwert reihe[i]


min := reihe[0]


max := reihe[0]

s := reihe[0]

Für i := 1 bis 149 tue

s := s + reihe[i]

reihe[i] < min	
ja	nein
min := reihe[i]	

reihe[i] > max	
ja	nein
max := reihe[i]	

d := s / 150

Ausgabe: Durchschnitt d

Ausgabe: Minimum min

Ausgabe: Maximum max

## Romz.mod

### Römische Zahlen

Eingabe: Zeichenkette rz  
(\*Römische Zahl\*)

s := 0 h1 := 0 h2 := 0  
l := Länge von rz

Für i := 0 bis (l-1) tue

rz[i]							
Fall 1	Fall 2	Fall 3	Fall 4	Fall 5	Fall 6	Fall 7	sonst
'I': h1:=1	'V': h1:=5	'X': h1:=10	'L': h1:=50	'C': h1:=100	'D': h1:=500	'M': h1:=1000	h1:=0

rz[i+1]							
Fall 1	Fall 2	Fall 3	Fall 4	Fall 5	Fall 6	Fall 7	sonst
'I': h2:=1	'V': h2:=5	'X': h2:=10	'L': h2:=50	'C': h2:=100	'D': h2:=500	'M': h2:=1000	h2:=0

h1 >= h2							
ja				nein			
s := s + h1				s := s - h1			

Ausgabe: Dezimalzahl s

## Stundenplan.mod

### Stundenplan

Für s := 0 bis 4 tue

Für z := 0 bis 9 tue

Eingabe: plan[z,s]  
(\* Fach für Tag (s+1) und Stunde (z+1) \*)

Ausgabe: Tabelle für Stundenplan

Für s := 0 bis 4 tue

Für z := 0 bis 9 tue

Ausgabe: plan[z,s]  
(\* Fach für Tag (s+1) und Stunde (z+1) \*)

## Textformat.mod

### Textformate

Eingabe: Zeichenkette text1

laenge := Länge( text1 )

Kopiere text1 nach text2

Ausgabe: 'linksbündig'

Ausgabe: text2

Für i:= 1 bis 60-laenge tue

Füge ein Leerzeichen  
in text2 an die 1.Stelle ein

Ausgabe: 'rechtsbündig'

Ausgabe: text2

Kopiere text1 nach text2

Für i:= 1 bis (60-laenge) DIV 2 tue

Füge ein Leerzeichen  
in text2 an die 1.Stelle ein

Ausgabe: 'zentriert'

Ausgabe: text2

## Tiersuche.mod

### Tiersuche

Initialisierung des Buchstabenschemas in text[z,s]

Ausgabe der Suchmatrix

Eingabe: Tiername tier

l := Länge(tier)

(\*Suche in Zeilen \*)

Für z := 0 bis 9 tue

Für s := 0 bis 10-l tue

Für i := 1 bis l tue

Füge text[z,s+i-1] in such ein

		such = tier	
ja			nein

Ausgabe: 'Gefunden!' Zeile z+1 Spalte s+1

such := ''

(\* Suche in Spalten analog \*)

Eingabe: weiter

Wdh.

bis weiter = 'N'

## Verbund.mod

### Verbund

minperson.alter := 200

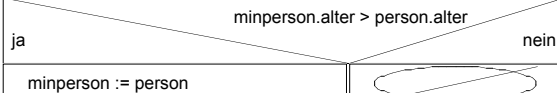
Eingabe: Anzahl der Personen  
anzahl

Für i:= 1 bis anzahl tue

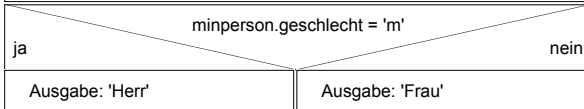
Eingabe: person.name

Eingabe: person.name

Eingabe: person.geschlecht



Ausgabe: 'Die jüngste Person ist:'



Ausgabe: minperson.name  
minperson.alter

## Wuerfel.mod

### Würfelsimulation

Für i:= 1 bis 1000 tue

x := Zufall(6) + 1

y := Zufall(6) + 1

auge[x+y-1] := auge[x+y-1] + 1

Für i:= 0 bis 11 tue

Ausgabe: Augenzahl i+10

Ausgabe: Anzahl der Würfe auge[i]

## Zahlenfolge.mod

### Zahlenfolge

Ausgabe: Partialsummenberechnung

Für i:= 0 bis 99 tue

folge[i] := 2 exp i

Ausgabe: folge[i]

Eingabe: n

s := 0

Für i:= 0 bis n tue

s := s + folge[i]

Ausgabe: Partialsumme s



# UNTERPROGRAMME

Modul	Struktogramm																				
BruchProz.mod	<div><div><b>BruchProz</b><table><tr><td>Eingabe: Bruch a</td><td></td></tr><tr><td>Ausgabe: Bruch a</td><td></td></tr><tr><td>Eingabe: Bruch b</td><td></td></tr><tr><td>Ausgabe: Bruch b</td><td></td></tr><tr><td>Addiere a und b --&gt; c</td><td></td></tr><tr><td>Kürze c</td><td></td></tr><tr><td>Ausgabe: Bruch c</td><td></td></tr></table></div><div><b>Prozedur Eingabe (VAR ein: tbruch)</b><table><tr><td>Eingabe: ein.zaehler</td></tr><tr><td>Eingabe: ein.nenner</td></tr></table></div><div><b>Prozedur Ausgabe (aus : tbruch)</b><table><tr><td>Ausgabe: aus.zaehler</td></tr><tr><td>Ausgabe: aus.nenner</td></tr></table></div><div><b>Prozedur addiere (x,y:tbruch; VAR z:tbruch)</b><table><tr><td>z.nenner := x.nenner * y.nenner</td></tr><tr><td>z.zaehler := x.zaehler * y.nenner + y.zaehler* x.nenner</td></tr></table></div></div>	Eingabe: Bruch a		Ausgabe: Bruch a		Eingabe: Bruch b		Ausgabe: Bruch b		Addiere a und b --> c		Kürze c		Ausgabe: Bruch c		Eingabe: ein.zaehler	Eingabe: ein.nenner	Ausgabe: aus.zaehler	Ausgabe: aus.nenner	z.nenner := x.nenner * y.nenner	z.zaehler := x.zaehler * y.nenner + y.zaehler* x.nenner
Eingabe: Bruch a																					
Ausgabe: Bruch a																					
Eingabe: Bruch b																					
Ausgabe: Bruch b																					
Addiere a und b --> c																					
Kürze c																					
Ausgabe: Bruch c																					
Eingabe: ein.zaehler																					
Eingabe: ein.nenner																					
Ausgabe: aus.zaehler																					
Ausgabe: aus.nenner																					
z.nenner := x.nenner * y.nenner																					
z.zaehler := x.zaehler * y.nenner + y.zaehler* x.nenner																					

	<div> <div> <b>Prozedur kuerze</b> </div> <div> <pre> t := 1 k.zaehler &gt; k.nenner ja      nein h := k.nenner      h := k.zaehler Für i := 1 bis h tue   k.nenner MOD i = 0   UND   k.zaehler MOD i = 0   ja      nein   t := i   k.nenner := k.nenner DIV t   k.zaehler := k.zaehler DIV t </pre> </div> </div>
Ellipseproz.mod	<div> <div> <b>Prozedur Ellipse (xm,ym,h,n: INTEGER)</b> </div> <div> <pre> Für x := -h bis +h tue   y := n * Wurzel(1-((x*x)/(h*h)))   Zeichne Punkt (xm+x , ym+y)   Zeichne Punkt (xm+x , ym-y) </pre> </div> </div>
Grf_proz.mod	<div> <div> <b>Ganzrationale Funktionen</b> </div> <div> <pre> Ausgabe: 'Ganzrationale Funktionen' 'f(x) = a(n)x^n + a(n-1)x^(n-1) + ... + a(1)x + a(0) ' eingeben ableiten werte_ber zeichnen </pre> </div> </div>

	<div> <b>Prozedur eingeben</b> <div> Eingabe: Grad der Funktion n </div> <div> Für i := n abwärts bis 0 tue <div> Eingabe: Koeffizient a[ i ] </div> </div> <div> Ausgabe der Funktionsgleichung </div> </div> <div> <b>Prozedur ableiten</b> <div> Für i := 0 bis n-1 tue <div> <math>b[i] := (i + 1) * a[i + 1]</math> </div> </div> <div> Ausgabe: Ableitungsfunktion </div> </div>
Kreisproz.mod	<div> <b>Prozedur Kreis (xm, ym, r: INTEGER)</b> <div> Für i := 0 bis r tue <div> <div> x := i y := Wurzel( r<sup>2</sup> - x<sup>2</sup>) </div> <div> Zeichne Punkt (xm + x, ym + y) </div> <div> Zeichne Punkt (xm + x, ym - y) </div> <div> Zeichne Punkt (xm - x, ym + y) </div> <div> Zeichne Punkt (xm - x, ym - y) </div> </div> </div> </div>
Kugel_F.mod	<div> <b>Kugelberechnung mit Funktionen</b> <div> Eingabe: Radius r </div> <div> Durchmesser berechnen ( r ) </div> <div> Oberfläche berechnen ( r ) </div> <div> Volumen berechnen ( r ) </div> </div>

### Funktion Durchmesser (rad: INTEGER): INTEGER

Rückgabe:  $2 * \text{rad}$

### Funktion Oberfläche (rad: INTEGER): INTEGER

$h := 4 * 3.14159 * \text{rad}^2$

Rückgabe: h

### Funktion Volumen (rad: INTEGER): INTEGER

$h := (4 * 3.14159 * \text{rad}^3) / 3$

Rückgabe: h

NFakultaet.mod

### Funktion Fakultät (n: INTEGER): LONGINT

f := 1

Für i := 1 bis n tue

f := f \* i

Rückgabe: f

Rechteck.mod

### Rechteck

Eingabe: linke untere x-Koord. xu

Eingabe: linke untere y-Koord. yu

Eingabe: rechte obere x-Koord. xo

Eingabe: rechte obere y-Koord. yo

Eingabe: Breite breite

Rechtecke zeichnen (xu, yu, xo, yo, breite)

### Prozedur Rechtecke\_zeichnen (x1, y1, x2, y2, br)

Für i := 0 bis br tue

Zeichne Rechteck (x1+i, y1+i, x2-i, y2-i)

Potenz.mod

### Funktion Potenz (basis: REAL; exponent: INTEGER): REAL

pot := 1

Für i := 1 bis exponent tue

pot := pot \* basis

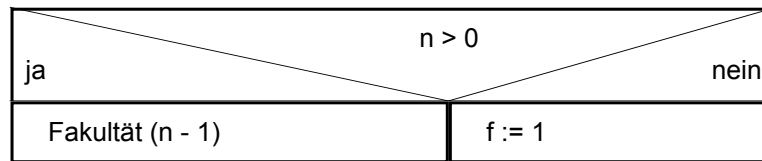
Rückgabe: pot

# REKURSION

Modul	Struktogramm
Ausgabe.mod	<p><b>Prozedur ausgeben (i: INTEGER)</b></p> <pre> graph TD     Start([ ]) --&gt; Cond{i = 0}     Cond -- ja --&gt; Out1[Ausgabe: i]     Out1 --&gt; End([ ])     Cond -- nein --&gt; Rec[ausgeben(i - 1)]     Rec --&gt; Out2[Ausgabe: i]     Out2 --&gt; End           </pre>
Dual.mod	<p><b>Prozedur trans( nzahl: INTEGER)</b></p> <pre> graph TD     Init[rest := 0] --&gt; Cond{nzahl &gt; 0}     Cond -- ja --&gt; Calc1[rest := nzahl MOD 2]     Calc1 --&gt; Calc2[nzahl := nzahl DIV 2]     Calc2 --&gt; Rec[trans(nzahl)]     Rec --&gt; End([X])     Cond -- nein --&gt; End     End --&gt; Out[Ausgabe: rest]           </pre>
Eins.mod	<p><b>Funktion rekursiv( x, y: INTEGER): INTEGER</b></p> <pre> graph TD     Cond{y &gt; 0} -- ja --&gt; IncX[x := x + 1]     IncX --&gt; DecY[y := y - 1]     DecY --&gt; Rec[rekursiv(x, y)]     Rec --&gt; Out[Rückgabe: x]     Cond -- nein --&gt; Out           </pre>

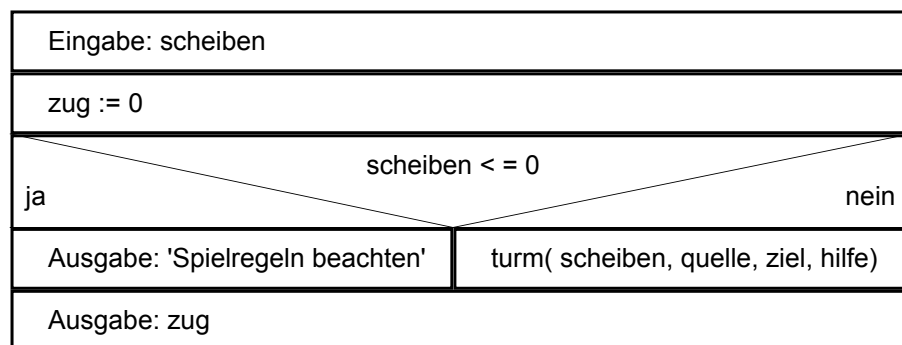
Fakult.mod

### Funktion Fakultät (n: INTEGER): INTEGER

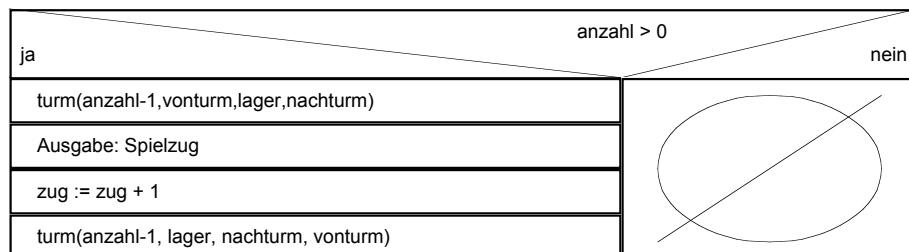


Hanoi.mod

### Türme von Hanoi



### Prozedur turm( anzahl: INTEGER; vonturm, nachturm, lager: CHAR)



Laby.mod

### Suche im Labyrinth

anz := 0

Labyrinth ausgeben

wege\_finden( 6, 12)

Ausgabe: 'Fertig!'

### Prozedur Marke\_setzen ( ze, sp: INTEGER; m: CHAR)

lab [ze, sp] := m

Ausgabe: m an die Position ze + 5 / sp + 5

### Funktion Rand\_erreicht( ze, sp: INTEGER): BOOLEAN

ze = 0 ODER ze = z-1 ODER  
sp = 0 ODER sp = s - 1

ja

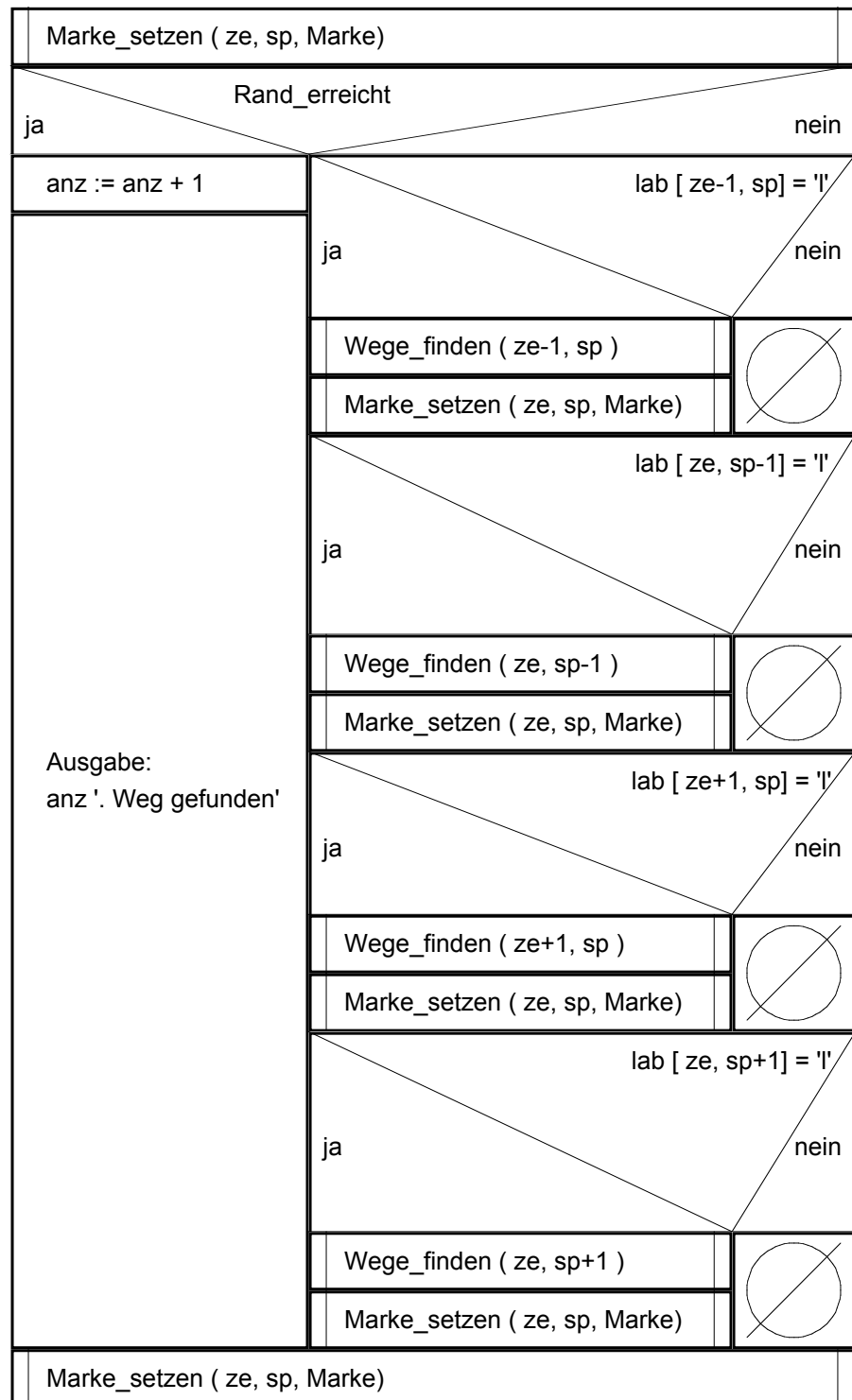
nein

Rückgabe: TRUE

Rückgabe: FALSE



## Prozedur Wege\_finden ( ze, sp: INTEGER )



N\_Damen.mod

## N\_Damen

Für i := 0 bis cdamen-1 tue

dame[ i ] := 0

zeile[ i ] := TRUE

Für i := 0 bis ( 2\*cdamen - 2 ) tue

d1[ i ] := TRUE

d2[ i ] := TRUE

anz := 0

dame\_setzen ( 0 )

Ausgabe: Anzahl der Lösungen anz

## Prozedur dame\_setzen( s: INTEGER)

Für z := 0 bis cdamen-1 tue

dnr1 := z

s > 0

ja

nein

dnr1 := dnr1 + s

dnr2 := z

s < cdamen-1

ja

nein

dnr2 := dnr2 + (cdamen-1-s)

zeile[ z ] & d1[ dnr1 ] & d2[ dnr2 ]

ja

nein

dame[ s ] := z

zeile[ z ] := FALSE

d1[ dnr1 ] := FALSE

d2[ dnr2 ] := FALSE

s < cdamen - 1

ja

nein

dame\_setzen( s + 1 )

ausgeben

zeile[ z ] := TRUE

d1[ dnr1 ] := TRUE

d2[ dnr2 ] := TRUE

## Prozedur ausgeben

anz := anz + 1

Für i := 0 bis cdamen -1 tue

Ausgabe: dame[ i ]

Rekbaum.mod

## REKBAUM

$l := 0$

Eingabe: Rekursionstiefe  $rt$

Baum( 300, 20, 300, 120,  $l$  )

## Prozedur Baum( $x_1, y_1, x_2, y_2, l$ : INTEGER)

$l := l + 1$

Zeichne eine Linie von ( $x_1 \mid y_1$ ) nach ( $x_2 \mid y_2$ )

nachrechts(  $x_1, y_1, x_2, y_2, x_3, y_3$  )

$l < rt$

ja

nein

Baum(  $x_2, y_2, x_3, y_3, l$  )

nachlinks(  $x_1, y_1, x_2, y_2, x_3, y_3$  )

$l < rt$

ja

nein

Baum( $x_2, y_2, x_3, y_3, l$ )

## Prozedur nachlinks( $x_1, y_1, x_2, y_2$ :INTEGER;VAR $x_3, y_3$ :INTEGER)

$nx := y_1 - y_2$     $ny := x_2 - x_1$     $bn := \text{Wurzel}(nx?+ny?)$

$b1 := \text{Wurzel}( (x_2-x_1)? + (y_2-y_1)? )$

$x_3 := x_2 + k*(x_2-x_1) - (k*b1*(y_1-y_2) / bn)$

$y_3 := y_2 + k*(y_2-y_1) - (k*b1*(x_2-x_1) / bn)$

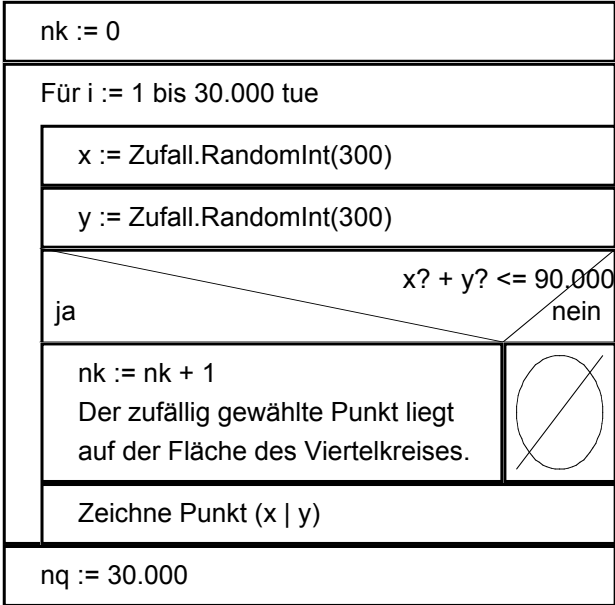
Zeichne eine Linie von ( $x_2 \mid y_2$ ) nach ( $x_3 \mid y_3$ )

	<div> <b>Prozedur nachrechts(x1,y1,x2,y2:INTEGER;VAR x3,y3:INTEGER)</b> <div> <math display="block">nx := y1 - y2 \quad ny := x2 - x1 \quad bn := \text{Wurzel}(nx^2 + ny^2)</math> <math display="block">b1 := \text{Wurzel}((x2 - x1)^2 + (y2 - y1)^2)</math> <math display="block">x3 := x2 + k \cdot (x2 - x1) - (k \cdot b1 \cdot (y1 - y2) / bn)</math> <math display="block">y3 := y2 + k \cdot (y2 - y1) - (k \cdot b1 \cdot (x2 - x1) / bn)</math>           Zeichne eine Linie von (x2   y2) nach (x3   y3)         </div> </div>
--	---

## ANDERES

Modul	Struktogramm
Monte.mod	<div> <b>Monte-Carlo-Methode</b> <div>           Grafikmodus einstellen           <div>Rahmen</div> <div>Zufall</div> <div>Tropfen</div> <math display="block">p := 4 \cdot nk / nq</math>           Ausgabe:            "Der Näherungswert von Pi beträgt:"            p         </div> </div> <div> <b>Prozedur Rahmen</b> <div>           Zeichne Quadrat mit            einbeschriebenen Viertelkreis         </div> </div>

**Prozedur Tropfen**



KonzGame.mod
--------------

## Konzentrationsspiel

Spielregeln	
richtig := 0	
falsch := 0	
Für Runde := 1 bis 10 tue	
Bildschirm löschen	
Cursor auf zufällige Position setzen	
zufällige Anzahl von Sternchen erzeugen: Zanzahl	
Für Stern := 1 bis Zanzahl tue	
Ausgabe: ' * '	
Eingabe: Anzahl, die der Spieler vermutet Sanzahl	
Zanzahl = Sanzahl	
ja	nein
Ausgabe: "Richtig!"	Ausgabe: "Falsch!"
richtig := richtig + 1	falsch := falsch + 1
Ausgabe: Auswertung des Spiels (richtig, falsch)	

## Prozedur Spielregeln

Ausgabe:  
"Auf dem Bildschirm erscheint an einer zufälligen Position eine Anzahl ( 1 - 9 ) von Sternchen ( \* ). Der Spieler soll diese Anzahl erkennen und im Anschluss eingeben. Dieser Vorgang wird 10x wiederholt und danach die Trefferquote ermittelt."

"Auf dem Bildschirm erscheint an einer zufälligen Position eine Anzahl ( 1 - 9 ) von Sternchen ( \* ). Der Spieler soll diese Anzahl erkennen und im Anschluss eingeben. Dieser Vorgang wird 10x wiederholt und danach die Trefferquote ermittelt."

