Eine *TI-Nspire* Bibliothek für die Trigonometrie

In diesem kleinen Aufstaz wollen wir mit dem *Nspire* eine Bibliothek mit dem Namen `trigo` zusammenstellen, die zur Lösung von trigonometrischen Grundaufgaben, dh., vor allem zur Auflösung von allgemeinen Dreiecken herangezogen werden kann. Die dort bereit gestellten Programme sollen mehrere Kenntnisstufen der Schülerinnen und Schüler berücksichtigen.

In den Notes steht die Beschreibung der Programme, die zuerst natürlich geschaffen, aber dann angeboten werden:

Es folgt eine Kopie der Notes

Bibliothek TRIGO

Sinus− und Kosinussatz

cosw(s1,s2,s3) berechnet den Winkel, der der Seite s2 gegenüberliegt, wenn die drei Seiten s1, s2, und s3 gegeben sind.

coss(s1,w3,s2) berechnet die dritte Seite s3, wenn zwei Seiten s1 und s2 und der von ihnen eingeschlossene Winkel w3 gegeben sind.

sins(s1,w1,w2) berechnet die Seite s2, die dem Winkel w2 gegenüberliegt, wenn ein korrespondierendes Paar s1, w1 gegeben sind.

sinw(s1,s2,w1) berechnet den Winkel w2, der der Seite s2 gegenüberliegt, wenn das Paar s1 und gegenüberliegender Winkel w1 gegeben sind. Für den Fall, dass der Winkel der kleineren Seite gegenüberliegt, werden beide möglichen Winkel ausgegeben − falls sie existieren.

Auflösung von Dreiecken,

wenn drei Bestimmungsstücke vorliegen und der Fall nach
SSS, SWS, WSW, SWW oder SSW angegeben wird. Auch der Fall WWW wird behandelt. Hier werden die Seitenlängen mit einem Proportionalitätsfaktor ausgegeben. In den Funktionen wird auf die Unterprogramme von oben zurückgegriffen.

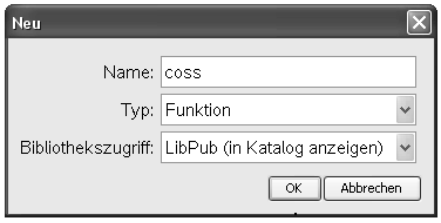
sss(s1,s2,s3)
sws(s1,w3,s2)
wsw(w1,s3,w2)
sww(s1,w1,w2)
ssw(s1,s2,w1)
www(w1,w2,w3)

liefern eine Matrix, die in der ersten Zeile die Seiten und in der zweiten Zeile die den Seiten gegenüberliegenden Winkel ausgibt.

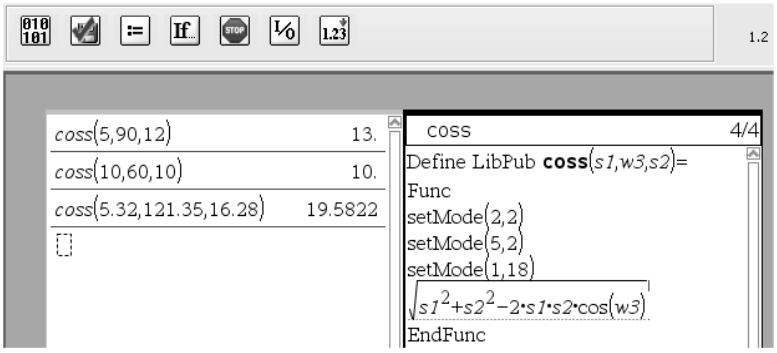
Das Programm **trigo(s1,s2,s3,w1,w2,w3)** liefert die Auflösung des Dreiecks, wenn drei der Parameter mit Zahlen belegt sind. In diesem Programm (eigentlich wieder Funktion) wird auf die "Unterprogramme" **sss** bis **www** zurückgegriffen.

Ich werde hier nicht alle Programme des Pakets *trigo* erzeugen, sondern einige davon Ihnen zum Programmieren überlassen. An Hand der vorgestellten Programme sollte Ihnen das nicht schwer fallen. Das komplette Paket wird Ihnen aber gerne zur Verfügung gestellt.

Öffnen Sie bitte ein neues Dokument und speichern Sie es sofort unter dem Namen *trigo* – oder unter einem anderen Namen, der Ihnen besser gefällt – im Verzeichnis *Eigene Dateien\TI-Nspire\mylib*. Für unsere Zwecke werden wir mit Funktionen auskommen, da keine Notwendigkeit besteht, globale Variable einzuführen. Dann öffnen Sie aus dem Calculator den Programmeditor zur Erstellung einer neuen Funktion und machen sie allgemein zugängig und damit auch im Katalog auffindbar.

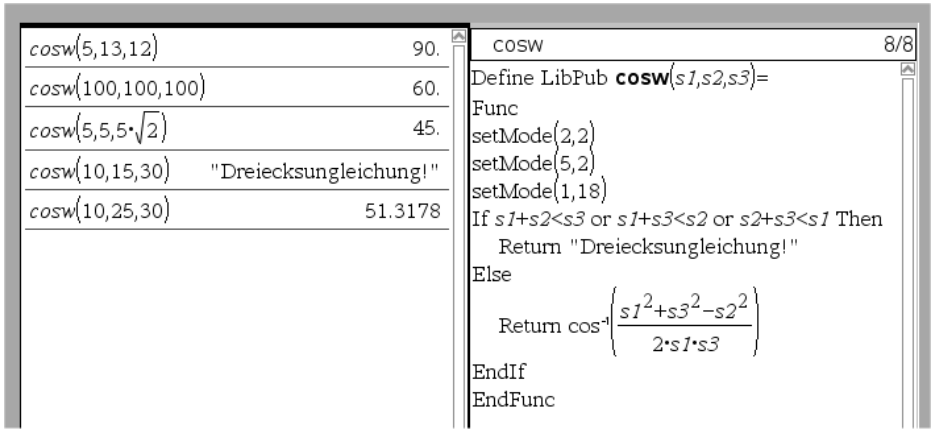


Die erste Funktion *coss* ist rasch geschrieben und auch getestet. Sie soll jene Seite berechnen, die dem Winkel *w3*, der von den Seiten *s1* und *s2* eingeschlossen wird, gegenüber liegt – der Kosinussatz wird eingefordert.



Über die drei SetMode-Anweisung, die wir über den letzten Button (1.23) erhalten, werden jene Einstellungen festgelegt, die unabhängig von den getroffenen Systemeinstellungen für die Ausführung dieser einen Funktion gelten sollen: Rechnen im Gradmaß, eine approximierte Ausgabe die mit 4 Dezimalstellen erfolgen soll. Damit werden die Grundeinstellungen von *Nspire* nicht verändert!
Die übrigen Schaltflächen dienen der raschen Eingabe der wichtigsten Programmierbefehle. Klicken Sie die Schaltflächen einfach an und sehen Sie sich die angebotenen Unterpunkte an.

Bei der Erstellung von *cosw*(*s*1,*s*2,*s*3) werden wir eine Überprüfung der Sinnhaftigkeit der Eingabe einbauen, denn die Seiten müssen die Dreiecksungleichung erfüllen.



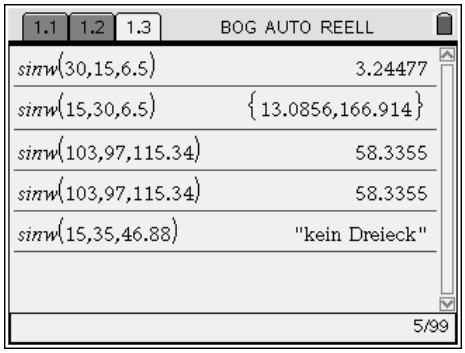
Beide *Return*-Anweisungen müssen nicht geschrieben werden, sie machen aber den Code leichter lesbar.

Jetzt fehlt uns noch der Sinussatz in seinen beiden Formulierungen: Zwei Winkel und eine zugehörige Seite sind gegeben und die zweite Seite *s*2 ist gesucht: *sins*(*s*1,*w*1,*w*2) und die heiklere Sache mit zwei Seiten und einem gegenüberliegenden Winkel, denn hier kann es eine, zwei oder gar keine Lösung geben.

Ich überlasse Ihnen gerne das Programmieren von *sins*(*s*1,*w*1,*w*2) und beschränke mich auf *sinw*(*s*1,*s*2,*w*1):

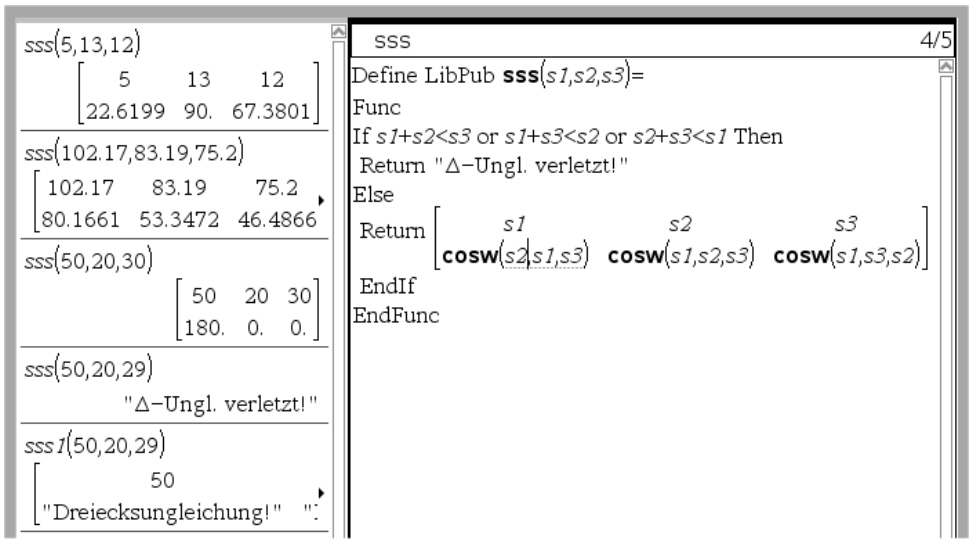
```
sinw                                  1(
Define LibPub sinw(s1,s2,w1)=
Func
Local h
setMode(5,2):setMode(1,18):setMode(2,2)
h:=s2·sin(w1)/s1
If h>1 Then
  Return "kein Dreieck"
  Else
  If s1≥s2 Then
      Return sin⁻¹(h)
    Else
      Return {sin⁻¹(h),180−sin⁻¹(h)}
  EndIf
EndIf
EndFunc
```

Beachten Sie bitte die geschachtelte If-Anweisung.



Nun haben wir die Grundformen von Sinus- und Kosinussatz als Funktionen definiert und können eine Stufe höher steigen.

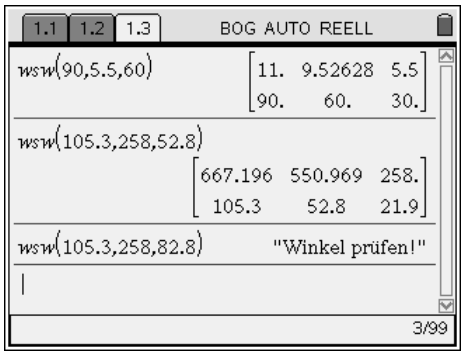
Der einfachste Fall liegt wohl vor, wenn alle drei Seiten gegeben sind:



Die Abfrage im Zusammenhang mit der Dreiecksungleichung führen wir nochmals durch. Wie es aussehen würde, wenn wir das der „Unterfunktion“ *cosw* überlassen, sehen sie im Calculatorfenster unter *ssw1*(50,20,29).

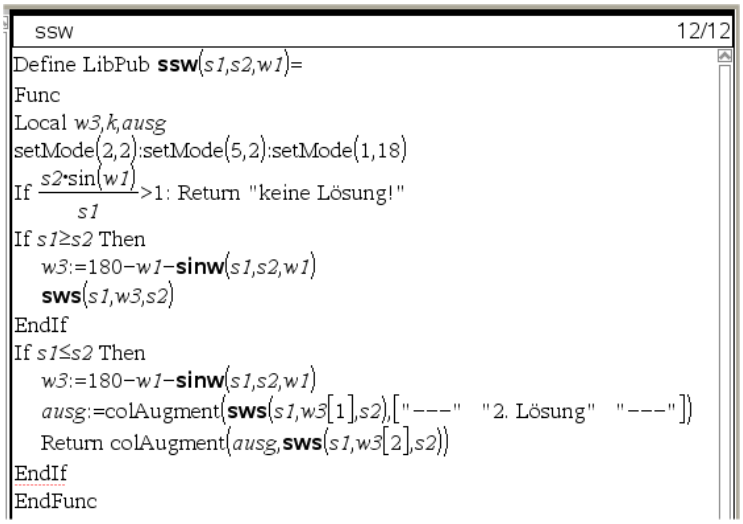
sws(s1,w2,s3) ist ganz einfach. Das kann ich getrost Ihnen überlassen. Sie müssen nur zuerst die Seite s2 mit *coss* berechnen (Vergessen Sie bitte nicht s2 als lokale Variable zu definieren) und dann sind sie ja schon wieder beim Fall SSS!

Der Fall wsw(*w*1,*s*3,*w*2) ist auch nicht schwierig. Nach Berechnung des dritten Winkels, können über zweimalige Anwendung von *sins* die restlichen beiden Seiten leicht gefunden werden. Die Ausgabematrix ist zu definieren. Eine Plausibilitätskontrolle ist vorzusehen. Betrachten Sie bitte den folgenden Schirm des Taschenrechners.

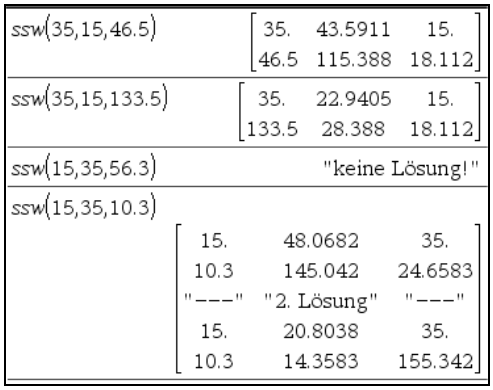


Auch der Fall *sww*(*s*1,*w*1,*w*2) lässt sich problemlos lösen, nachdem der fehlende Winkel *w*3 gefunden wurde, kann auf *wsw*(*w*3,*s*1,*w*2) zurückgegriffen werden.

Problematisch ist wiederum nur der Fall, wenn zwei Seiten und der einer Seite gegenüberliegende Winkel vorliegen, wobei die Parameter in einer eindeutigen Weise eingegeben werden müssen *ssw*(*s*1,*s*2,*w*1). Hier tritt wieder die Fallunterscheidung auf:



Und hier sind einige Testdreiecke:



Vergessen Sie bitte nicht, immer wieder zu speichern. Jetzt wäre es schon längst an der Zeit, die Funktionalität der Bibliothek zu überprüfen.

Öffnen Sie bitte ein neues Dokument und rufen Sie den Calculator auf.

Wir wollen das Dreieck ABC mit a = 105,33, b = 82,17 und α = 110,13° nach seinen restlichen Bestimmungsstücken auflösen. Wir erkennen, dass der Fall SSW angesprochen ist.

Aktualisieren Sie zuerst die Bibliotheken und öffnen Sie anschließend den Katalog.

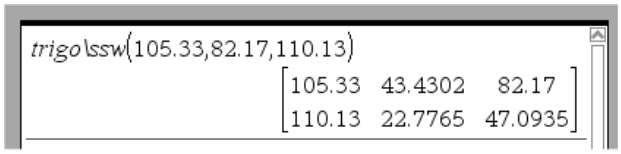
Unter Option 6 wird Ihnen Ihre eigene Bibliothek *trigo* angeboten. Wenn Sie auf das +-Zeichen klicken, sehen Sie alle darin enthaltenen Programm und Funktionen.

Doppelklicken Sie nun auf *ssw*.

Im Calculatorfenster sehen Sie nun den Funktionsaufruf eingetragen und Sie können die erforderlichen Parameter eintragen.

Wir erwarten eine Lösung.

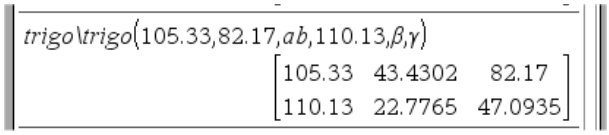
Es liegt am Benutzer, die Lösung richtig zu interpretieren, wobei das hier nicht schwierig sein sollte.



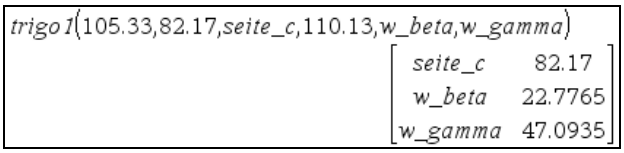
In der ersten Zeile stehen die drei Seiten und darunter die jeweils gegenüber liegenden Winkel. In der Standardbezeichnung ist demnach c = 43,43, β = 47,09° und γ = 22,78°. Wir erkennen, dass die Funktion, wie auch alle anderen bisher definierten, von überall her recht bequem aufgerufen werden können.

Jetzt hätten wir aber noch einen Fall vergessen: was tun, wenn drei Winkel vorliegen. Ich weiß, dass dies meist übergangen wird, aber es ist doch ein interessanter Fall. Hier gibt es unendlich viele Lösungsdreiecke, die alle zueinander ähnlich sind. Das werden wir auch –am Ende des Kapitels – noch näher behandeln.

So weit, so gut. Aber das Bessere ist des Guten Feind. Es wäre doch noch einfacher, wenn wir die Auswahl der richtigen Funktion auch dem Computer überlassen könnten! Dann könnte die Durchführung so aussehen:



Oder vielleicht noch freundlicher? Was sagen Sie dazu:



Hier will ich nicht die ganze Prozedur nachvollziehen. Es geht zuerst einmal darum, aus den vorliegenden Daten herauszufiltern, welcher der Kongruenzsätze anzuwenden ist. Dann sind fallweise die Seiten oder Winkel zu vertauschen, dass sie auch richtig verarbeitet werden.

Für die Klassifikation habe ich mir ein Schema zurecht gelegt, das jedem möglichen Angabefall eine eindeutige Zahl zuordnet.

Nach den Regeln der Kombinatorik kann man auf 20 verschiedene Arten 3 Elemente – ohne Wiederholung – aus 6 auswählen, weil $\binom{6}{3} = 20$.

	A	B a	C b	D c	E α	F β	G γ	H
1	www	0	0	0	1	1	1	7
2	sww	0	0	1	0	1	1	11
3	sww	0	0	1	1	0	1	13
4	wsw	0	0	1	1	1	0	14
5	sww	0	1	0	0	1	1	19
6	wsw	0	1	0	1	0	1	21
7	sww	0	1	0	1	1	0	22
8	ssw	0	1	1	0	0	1	25
9	ssw	0	1	1	0	1	0	26
10	sws	0	1	1	1	0	0	28
11	wsw	1	0	0	0	1	1	35
12	sww	1	0	0	1	0	1	37
13	sww	1	0	0	1	1	0	38
14	ssw	1	0	1	0	0	1	41
15	sws	1	0	1	0	1	0	42
16	ssw	1	0	1	1	0	0	44
17	sws	1	1	0	0	0	1	49
18	ssw	1	1	0	0	1	0	50
19	ssw	1	1	0	1	0	0	52
20	sss	1	1	1	0	0	0	56
21								

Diese Tabelle habe ich auch gleich mit *Nspire* angelegt. Fügen Sie eine Seite *Lists & Spreadsheet* ein, tragen Sie die Überschriften und Daten in die Spalten *A* bis *G* ein. In die Zelle *H1* schreiben Sie bitte die Formel =32*b1+16*c1+8*d1+4*e1+2*f1+g1 und kopieren diese – wie in Excel in die restlichen Zellen der Spalte *H*.

Leider finde ich kein deutliches Muster, das ich für eine effiziente Programmierung ausnützen könnte.

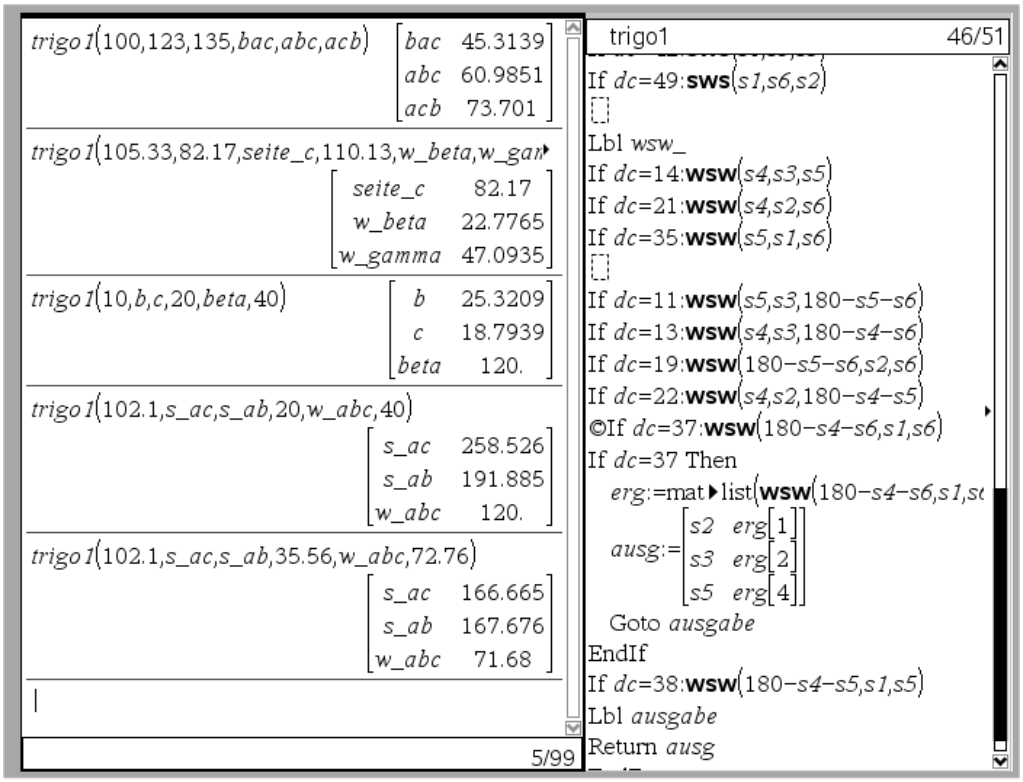
Daher mache ich es auf die „einfachste“ Art und Weise, wie dann der nächste Programmausschnitt zeigt.

In der Zeile, die mit v[1,i] beginnt, werden den Elementen des Vektors *v* die Werte 1 oder 0 zugewiesen, je nachdem, ob der Eingabewert mit einer Ziffer oder dem Dezimalpunkt beginnt (ASCII-Code <=57) oder nicht. Damit erhalte ich die Zeilen der obigen Tabelle. Wenn die Summe der Elemente nicht mit 3 übereinstimmt, liegt ein Eingabefehler vor. Dieser Vektor wird skalar mit dem Vektor [32, 16, 8, 4, 2, 1] multipliziert und führt so zum Code jedes einzelnen Falles. Dann wird nach jedem Code einzeln verfahren, wobei einige zuerst gruppenweise zusammengefasst und dann in den durch Einsprungmarken (*lbl*) gekennzeichneten Programmteilen einzeln behandelt werden. Ich glaube, dass Sie das leicht nachvollziehen können. Es ist auch eine interessante und Gewinn bringende Aufgabe für Schülerinnen und Schüler, diese Fälle zu analysieren.

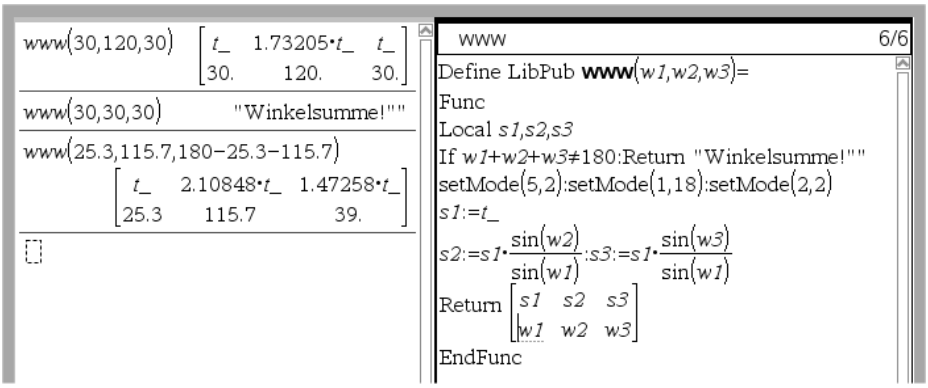
```
trigo                                                    0/21
Define LibPub trigo(s1,s2,s3,s4,s5,s6)=
Func
Local v,i,dc
v:=[0  0  0  0  0  0]
For i,1,6
v[1,i]:=when(ord(left(string(#("s"&string(i))))))≤57,1,0)
EndFor
If sum(mat▸list(v))≠3:Return "Eingabefehler!"
dc:=dotP(v,[32  16  8  4  2  1])
If dc=56:sss(s1,s2,s3):If dc=7:www(s4,s5,s6)
If dc=28 or dc=42 or dc=49:Goto sws_
If dc=14 or dc=21 or dc=35:Goto wsw_
If dc=11 or dc=13 or dc=19 or dc=22 or dc=37 or dc=38:Goto wsw_
© Fall SSW
If dc=52:ssw(s1,s2,s4):If dc=44:ssw(s1,s3,s4):If dc=26:ssw(s2,s3,s5)
If dc=50:ssw(s2,s1,s5):If dc=41:ssw(s3,s1,s6):If dc=25:ssw(s3,s2,s6)
Lbl sws_
If dc=28:sws(s2,s4,s3):If dc=42:sws(s1,s5,s3):If dc=49:sws(s1,s6,s2)
Lbl wsw_
If dc=14:wsw(s4,s3,s5):If dc=21:wsw(s4,s2,s6):If dc=35:wsw(s5,s1,s6)
If dc=11:wsw(s5,s3,180−s5−s6):If dc=13:wsw(s4,s3,180−s4−s6)
If dc=19:wsw(180−s5−s6,s2,s6):If dc=22:wsw(s4,s2,180−s4−s5)
If dc=37:wsw(180−s4−s6,s1,s6):If dc=38:wsw(180−s4−s5,s1,s5)
9 EndFunc
```

Im nächsten Bild zeige ich eine Möglichkeit, die Ausgabe noch angenehmer für den Anwender zu gestalten. Für jeden Fall wird die spezifische Ausgabematrix gestaltet und nach dem Label *ausgabe* im Calculatorfenster ausgegeben.

Beachten Sie den Einsatz des *indirection*-Kommandos # zur programminternen Erzeugung einer Variablen. Sie finden mehr dazu in den Referenzen.



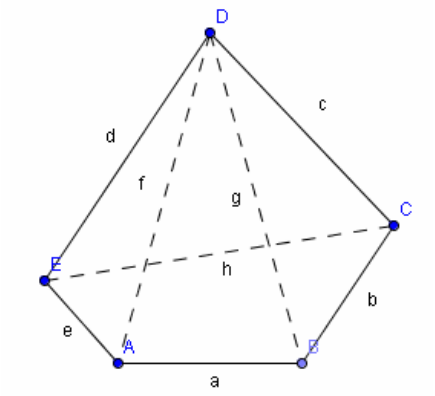
Jetzt bin ich Ihnen noch den Fall WWW schuldig! Ich führe den Proportionalitätsfaktor *t_* ein und gebe die Seiten mit diesem Faktor behaftet aus.



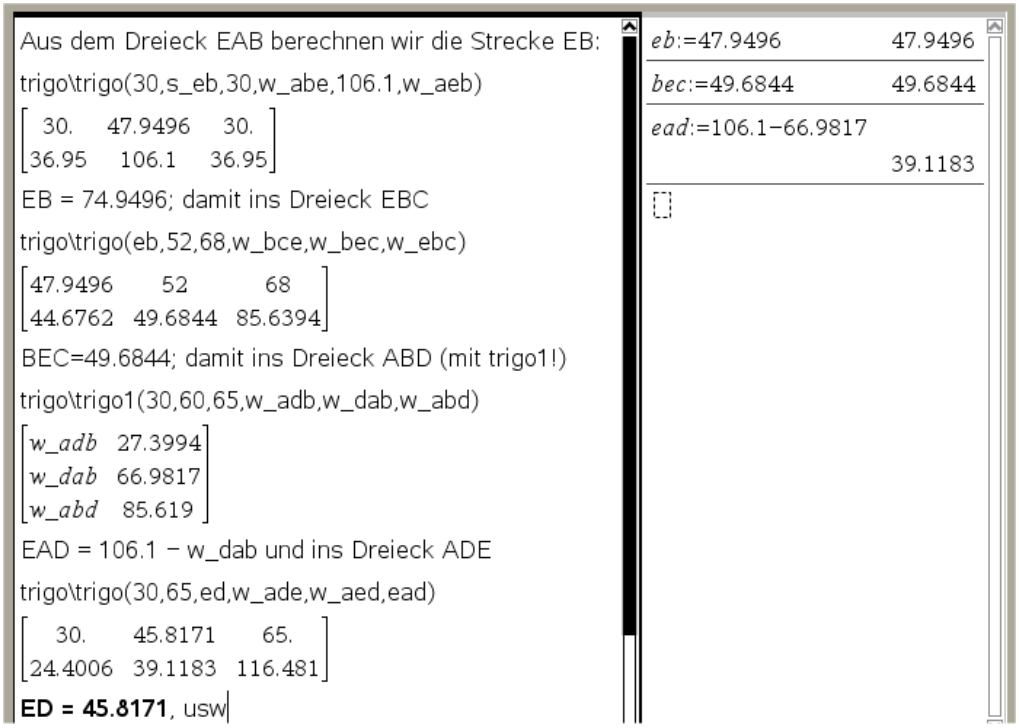
Zum Abschluss lösen wir mit dem Werkzeug eine komplexere „traditionelle Vermessungs-aufgabe“:

Von einem Fünfeck ABCDE kennt man:
AB = 30, BC = 52, AE = 30, AD = 65,
BD = 60, EC = 68
und den Winkel EAB = 106,1°.

Berechne ED und CD.



Wir bearbeiten die Aufgabe gleich in den „frischen“ Notes und evaluieren darin die Funktionen. Auch die Zuweisungen *eb*:=, *bec*:= könnten in den Notes erfolgen.



Josef Böhm, nojo.boehm@pgv.at