Das Dreieck

C++ Qt5-Übung

2010

Nah! Wie viele Dreiecke kannst du im obigen Bild erkennen? Opa sagt mit Löwenzahn einige tausend! Einigen wir uns auf vier. Dann ist da noch der Lothar Kusch! Ein Klassiker unter den Mathematikbüchern. Ein sehr gutes Buch zum Selbstlernen mit vielen Aufgaben und Lösungen. Das Buch stammt aus dem Jahr 1975. Die Wertschätzung unter den Technikern ist riesig. Wie oft ist das schon passiert! Da kommen Kinder, Enkel aus der Schule nach Hause mit schlecht leserlichen Aufgabenblätter (20 Cent pro Stück müssen sie dafür bezahlen) und eine Aufgabenlösung zur Selbstkontrolle fehlt. "Bitte, können Sie das Thema noch einmal kurz wiederholen"? Nee! Frage deinen älteren Bruder...!

Gar keine Frage, Dreiecke sind interessante Figuren. Zwei Anwendungen möchte ich hier kurz einflechten: Das "Försterdreieck" und das "Maurerdreieck". Förster und Waldarbeiter bestimmen auch heute noch die Höhe von Bäumen mit dem sogenannten "Försterdreick".
In der Regel handelt es sich um ein handliches gleichschenkliges Dreieck. Du findest im Internet unter *https://de.wikipedia.org/wiki/Försterdreieck* eine ausführliche Beschreibung. Das Maurerdreieck ist nicht so bekannt. Man kann ausgerüstet mit nur einem Zollstock den rechten Winkel bestimmen. Angenommen du baust eine Gartenlaube auf, so muss das Fundament schon exakt im rechten Winkel stehen. Hierzu wird von einer Ecke jeweils eine Strecke von 30 cm und 40 cm gemessen und markiert. Zwischen den beiden Markierungen wird nun die Hypotenuse ausgemessen. Diese muss genau 50 cm sein. Dann hast du einen rechten Winkel.

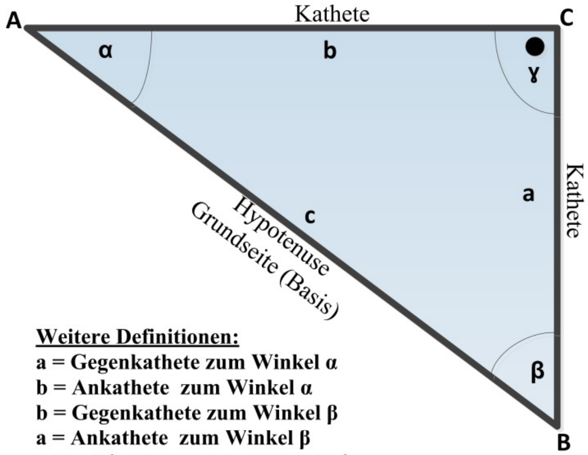
Das nachfolgend beschriebene Programm "DasDreieck" soll eine Hilfe sein. Für mich war es eine weitere gute Übung in der Programmiersprache C++ mit den Schwerpunkten "Klassen, Vererbung, und Grafiken". Im Internet gibt es viele Dreiecks-Rechner jedoch habe ich keines gefunden, welches Lösungswege aufzeigt und, wenn der Maßstab es erlaubt, eine grafische Visualisierung des Dreiecks zeigt. Zunächst beschäftigen wir uns mit den Grundlagen des Dreiecks (nur kurz). Danach werden die benötigten Formeln in einer Liste bzw. Tabelle gepackt. Ein kleiner Ausflug in die Kombinatorik ist auch mit dabei. Zum Schluss noch eine Mischung aus Bedienung des Programms und wie wurden die Kern-Algorithmen gelöst. Die Applikation "DasDreieck" ist folgendermaßen gegliedert:

- Eingabe der Variablen.
- Plausibilitätsprüfung der Eingaben.
- Berechnung.
- Gibt es eine zweite Lösung?
- Gibt es keine Lösung?
- Visualisierung des Lösungsweges.
- Visualisierung des Dreiecks als Grafik, Konstruktion.
- Drucken

Für diese Dokumentation benutze ich zwei Werkzeuge: Mikrosoft Word und Mikrosoft Visio. Da ich vieles parallel zur Programmierung protokolliert habe, einfach damit ich es nicht vergesse, habe ich sehr häufig das Schriftbild gewechselt. Na ja und nun habe ich keine Lust das zu korrigieren. Entschuldigung.

Allen Lesern wünsche ich viel Spaß und bitte um Nachsicht, wenn das eine oder andere nicht den „Geschmack“ trifft. Was ich hier tue fällt unter Hobbybastelei. Ich bin kein Profi.
Trotz aller Bemühungen lassen sich Fehler nie ganz vermeiden. Für Hinweise bin ich jedem dankbar.

Rechtwinkliges Dreieck



Weitere Definitionen:

a = Gegenkathete zum Winkel α
b = Ankathete zum Winkel α
b = Gegenkathete zum Winkel β
a = Ankathete zum Winkel β
γ = 90° (größte Seite c gegenüber)

$$180° = \alpha + \beta + \gamma$$

$$a^2 + b^2 = c^2$$

$$\sin\alpha = \frac{a}{c} \qquad \sin\beta = \frac{b}{c}$$

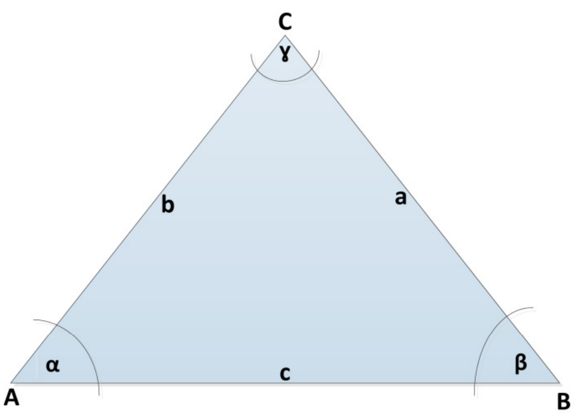
$$\cos\alpha = \frac{b}{c} \qquad \cos\beta = \frac{a}{c}$$

$$\tan\alpha = \frac{a}{b} \qquad \tan\beta = \frac{b}{a}$$

$$\cot\alpha = \frac{b}{a} \qquad \cot\beta = \frac{a}{b}$$

Nichtrechtwinkliges Dreieck

Schiefwinklige-, spitzwinklige- und stumpfwinklige Dreiecke



$$\frac{\sin\alpha}{a} = \frac{\sin\beta}{b} = \frac{\sin\gamma}{c}$$

$$a^2 = b^2 + c^2 - 2bc * \cos\alpha$$

$$b^2 = a^2 + c^2 - 2ac * \cos\beta$$

$$c^2 = a^2 + b^2 - 2ab * \cos\gamma$$

In jedem Dreieck ist die Summe zweier Seiten größer als die dritte Seite: a + b > c ; a + c > b ; b + c > a

In jedem Dreieck ist die Differenz zweier Seiten kleiner als die dritte Seite: a - b < c ; a - c < b ; c - b < a

In jedem Dreieck liegt dem größtem Winkel die größte Seite gegenüber und umgekehrt

Dreiecksarten:

Ungleichseitiges Dreieck, wenn alle drei Seiten verschieden lang sind: a ≠b ≠ c

Gleichschenkliges Dreieck, wenn zwei Seiten gleich lang sind: a = b ≠ c, α = β

Spitzwinkliges Dreieck, wenn alle Winkel spitz sind: α < 90°, β <90°, γ < 90°

Stumpfwinkliges Dreieck, wenn ein Winkel ein stumpfer ist: γ > 90°

Gleichseitiges Dreieck, wenn alle Seiten gleich lang sind: a = b = c, α = β = γ = 60°

Wie breit ist der Fluss?

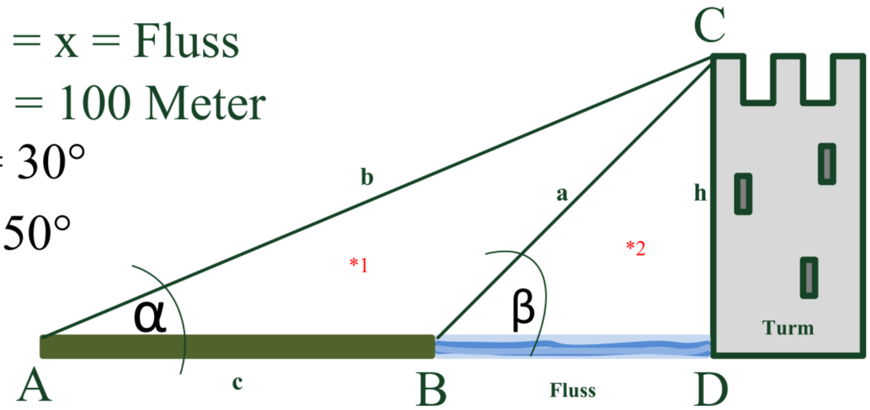
Beispiel für Pythagoras, Sinusfunktion und Sinussatz!

$\overline{BD} = x = \text{Fluss}$

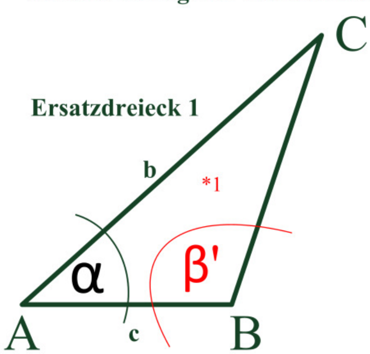
$\overline{AB} = 100 \text{ Meter}$

$\alpha = 30°$

$\beta = 50°$



Man erkennt zwei Dreiecke: (*1) stumpfwinkliges- und (*2) rechtwinkliges Dreieck. Vom stumpfwinkligen Dreieck können alle drei Winkel berechnet werden. Weiterhin kann mit Hilfe des Sinussatzes die Seite b und mit der Sinusfunktion (für rechtwinklige Dreiecke) die Höhe h berechnet werden. Es folgt ein Ersatzdreieck und die Berechnung:



$$\beta' = 180° - \beta = 180° - 50° = 130°$$

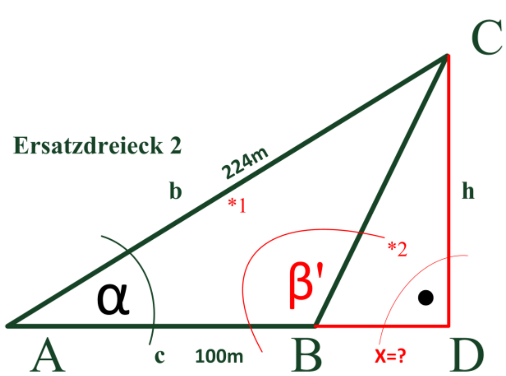
$$\gamma = 180° - \alpha - \beta' = 180° - 30° - 130° = 20°$$

Sinussatz:

$$\frac{b}{\sin \beta'} = \frac{c}{\sin \gamma} = \frac{b}{\sin 130°} = \frac{100}{\sin 20°}$$

$$b = \frac{100 * \sin 130°}{\sin 20°} = 224 \text{ Meter}$$

Als nächstes wird die Höhe h berechnet. Aus den zwei Dreiecken (*1) und (*2) wird jetzt ein einziges rechtwinkliges Dreieck geformt.



Sinusfunktion:

$$\sin \alpha = \frac{h}{b} \quad h = b * \sin \alpha = 224 \text{ m} * \sin 30°$$

Höhe h = 112 Meter

Zum Schluss, wird mittels Pythagoras die Breite des Flusses berechnet:

$$x + 100 = \sqrt{224^2 - 112^2}$$

$$x + 100 = 194$$

Der Fluss ist 94 Meter breit!

© Hans Busche

Kombinatorik / Permutation

https://de.wikipedia.org/wiki/Kombinatorik

Ein nichtrechtwinkliges Dreieck wird mit sechs Größen konstruiert. Zur Berechnung des nichtrechtwinkligen Dreiecks müssen drei Werte bekannt sein. Das Programm „DasDreieck“ muss alle Varianten abdecken. Wie viele Möglichkeiten gibt es? Ich habe mir einen Zettel genommen, dann eine Tabelle skizziert und alle Möglichkeiten eingetragen, diese dann zigmal kontrolliert. Anschließend hat es mich dann schon interessiert, wie man mit so etwas umgeht. Im Internet gibt es ein paar interessante Seiten zum Thema „Kombinatorik und Permutation“.

<u>Nichtrechtwinkliges Dreieck:</u>
Sechs Dreiecksgrößen sind sechs Objekte: n=6
Drei Auswahlmöglichkeiten: k = 3

$$\boldsymbol{Möglichkeiten = \frac{n!}{(n-k)!*k!} = \frac{6*5*4*3*2*1}{(6-3)!*3!} = \frac{720}{36} = 20}$$

!=Fakultät (4! = 4*3*2*1=24)

<u>Rechtwinkliges Dreieck:</u>
Ein Winkel ist immer 90°! Daher nur *<u>fünf</u>* Dreiecksgrößen.
Fünf Dreiecksgrößen sind fünf Objekte: n=5
Zwei Auswahlmöglichkeiten: k = 2

$$\boldsymbol{Möglichkeiten = \frac{n!}{(n-k)!*k!} = \frac{5*4*3*2*1}{(5-2)!*2!} = \frac{120}{12} = 10}$$

!=Fakultät (4! = 4*3*2*1=24)

Daraus folgt: In der C++ Applikation muss es eine „Switch/Case-Anweisung“ geben, die 20 Varianten abdeckt. Bei einem rechtwinkligen Dreieck, welches mit zwei bekannten Größen berechnet wird, kommt man auf 10 Möglichkeiten.

Auf den nachfolgenden Seiten sind die benötigten Formeln zur Berechnung des Dreiecks in Abhängigkeit der fehlenden Größen in einer Tabelle zusammengefasst. Es sind zwei Tabellen erforderlich: a) für nichtrechtwinklige- und b) für rechtwinklige Dreiecke. Anschließend folgen die Besonderheiten bestimmter Dreieckskonstruktionen.

Formel-Matrix für nichtrechtwinklige Dreiecke

● → gegeben!

Nichtrechtwinklige Dreiecke

	a	b	c	α	β	γ			
1	●	●	●				$\cos\alpha = \frac{b^2 + c^2 - a^2}{2bc}$	$\cos\beta = \frac{a^2 + c^2 - b^2}{2ac}$	$\cos\gamma = \frac{a^2 + b^2 - c^2}{2ab}$
2	●	●		●			$\sin\beta = \frac{\sin\alpha * b}{a}$	$\gamma = 180° - \beta - \alpha$	$c^2 = a^2 + b^2 - 2ab * \cos\gamma$
3	●	●			●		$\sin\alpha = \frac{\sin\beta * a}{b}$	$\gamma = 180° - \beta - \alpha$	$c^2 = a^2 + b^2 - 2ab * \cos\gamma$
4	●	●				●	$c^2 = a^2 + b^2 - 2ab * \cos\gamma$	$\cos\alpha = \frac{b^2 + c^2 - a^2}{2bc}$	$\cos\beta = \frac{a^2 + c^2 - b^2}{2ac}$
5	●		●	●			$\sin\gamma = \frac{\sin\alpha * c}{a}$	$\beta = 180° - \alpha - \gamma$	$b^2 = a^2 + c^2 - 2ac * \cos\beta$
6	●		●		●		$b^2 = a^2 + c^2 - 2ac * \cos\beta$	$\cos\alpha = \frac{b^2 + c^2 - a^2}{2bc}$	$\cos\gamma = \frac{a^2 + b^2 - c^2}{2ab}$
7	●		●			●	$\sin\alpha = \frac{\sin\gamma * a}{c}$	$\beta = 180° - \alpha - \gamma$	$b^2 = a^2 + c^2 - 2ac * \cos\beta$
8	●			●	●		$\gamma = 180° - \beta - \alpha$	$b = \frac{\sin\beta * a}{\sin\alpha}$	$c = \frac{\sin\gamma * a}{\sin\alpha}$
9	●			●		●	$\beta = 180° - \alpha - \gamma$	$b = \frac{\sin\beta * a}{\sin\alpha}$	$c = \frac{\sin\gamma * a}{\sin\alpha}$
10	●				●	●	$\alpha = 180° - \beta - \gamma$	$b = \frac{\sin\beta * a}{\sin\alpha}$	$c = \frac{\sin\gamma * a}{\sin\alpha}$
11		●	●	●			$a^2 = b^2 + c^2 - 2bc * \cos\alpha$	$\cos\beta = \frac{a^2 + c^2 - b^2}{2ac}$	$\cos\gamma = \frac{a^2 + b^2 - c^2}{2ab}$
12		●	●		●		$\sin\gamma = \frac{\sin\beta * c}{b}$	$\alpha = 180° - \beta - \gamma$	$a^2 = b^2 + c^2 - 2bc * \cos\alpha$
13		●	●			●	$\sin\beta = \frac{\sin\gamma * b}{c}$	$\alpha = 180° - \beta - \gamma$	$a^2 = b^2 + c^2 - 2bc * \cos\alpha$
14		●		●	●		$\gamma = 180° - \beta - \alpha$	$a = \frac{\sin\alpha * b}{\sin\beta}$	$c = \frac{\sin\gamma * a}{\sin\alpha}$
15		●		●		●	$\beta = 180° - \alpha - \gamma$	$a = \frac{\sin\alpha * b}{\sin\beta}$	$c = \frac{\sin\gamma * a}{\sin\alpha}$
16		●			●	●	$\alpha = 180° - \beta - \gamma$	$a = \frac{\sin\alpha * b}{\sin\beta}$	$c = \frac{\sin\gamma * a}{\sin\alpha}$
17			●	●	●		$\gamma = 180° - \beta - \alpha$	$a = \frac{\sin\alpha * c}{\sin\gamma}$	$b = \frac{\sin\beta * a}{\sin\alpha}$
18			●	●		●	$\beta = 180° - \alpha - \gamma$	$a = \frac{\sin\alpha * c}{\sin\gamma}$	$b = \frac{\sin\beta * a}{\sin\alpha}$
19			●		●	●	$\alpha = 180° - \beta - \gamma$	$a = \frac{\sin\alpha * c}{\sin\gamma}$	$b = \frac{\sin\beta * a}{\sin\alpha}$
20				●	●	●	***Unendlich viele Lösungen!***		

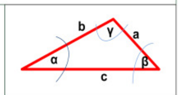
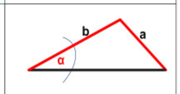
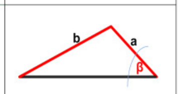
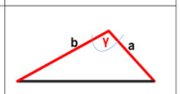
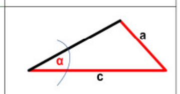
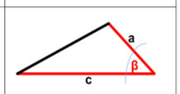
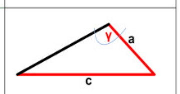
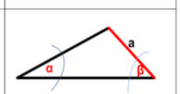
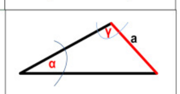
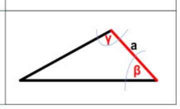
Formel-Matrix für rechtwinkliges Dreiecke

● → gegeben!

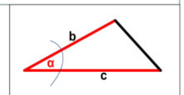
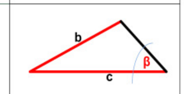
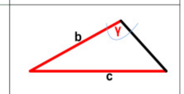
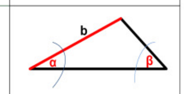
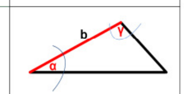
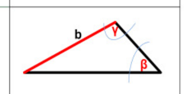
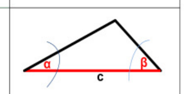
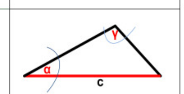
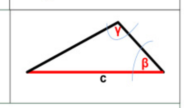
Rechtwinklige Dreiecke

	a	b	c	α	β			
1	●	●				$c=\sqrt{a^2+b^2}$	$\sin\alpha=\frac{a}{c}$	$\beta=180°-\gamma-\alpha$
2	●		●			$\sin\alpha=\frac{a}{c}$	$\beta=180°-\gamma-\alpha$	$b=c*\sin\beta$
3	●			●		$\beta=180°-\gamma-\alpha$	$c=\frac{a}{\sin\alpha}$	$b=c*\sin\beta$
4	●				●	$\alpha=180°-\gamma-\beta$	$c=\frac{a}{\sin\alpha}$	$b=c*\sin\beta$
5		●	●			$\sin\beta=\frac{b}{c}$	$\alpha=180°-\gamma-\beta$	$a=c*\sin\alpha$
6		●		●		$\beta=180°-\gamma-\alpha$	$c=\frac{b}{\sin\beta}$	$a=c*\sin\alpha$
7		●			●	$\alpha=180°-\gamma-\beta$	$c=\frac{b}{\sin\beta}$	$a=c*\sin\alpha$
8			●	●		$\beta=180°-\gamma-\alpha$	$b=c*\sin\beta$	$a=c*\sin\alpha$
9			●		●	$\alpha=180°-\gamma-\beta$	$b=c*\sin\beta$	$a=c*\sin\alpha$
				●	●	undefiniert		

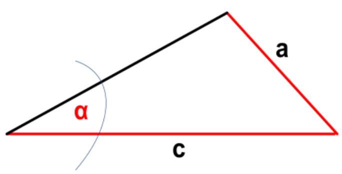
Fallunterscheidung Dreieckskonstruktion

switch case	a	b	c	α	β	γ	Konstruktion	
1	●	●	●				SSS	
2	●	●		●			SSW	
3	●	●			●		SSW	
4	●	●				●	SWS	
5	●		●	●			SSW	
6	●		●		●		SWS	
7	●		●			●	SSW	
8	●			●	●		SWW	
9	●			●		●	SWW	
10	●				●	●	WSW	

© Hans Busche

switch case	a	b	c	α	β	γ	Konstruktion	
11		●	●	●			SWS	
12		●	●		●		SSW	
13		●	●			●	SSW	
14		●		●	●		SWW	
15		●		●		●	WSW	
16		●			●	●	SWW	
17			●	●	●		WSW	
18			●	●		●	SWW	
19			●		●	●	SWS	
20				●	●	●	Unendlich viele Lösungen	

Fallunterscheidung! Betrachtet wird hier die Konstruktion 5, SSW. Gegeben sind: Seiten a ;c und der Winkel α.

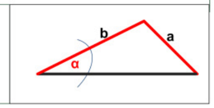


a>c	a=c	a<c
Eine Lösung	Eine Lösung	Eine, zwei oder keine Lösung

Die Software muss hier jeden Einzelfall prüfen. Der „*gibt es eine zweite Lösung -Algorithmus*" für das obige Beispiel 5, SSW sieht folgendermaßen aus: Zuerst werden alle fehlenden Größen berechnet. Der berechnete Winkel Beta β wird von 180° subtrahiert. Ist dieser positiv, gibt es zwei gültige Winkel β_1 und β_2. Danach wird $\gamma_2 = 180 - \alpha - \beta_2$ ausgerechnet. Ist der Winkel γ_2 positiv dann gibt es zwei Lösungen und die Applikation muss zur Berechnung der Lösung „2" eine weitere „Schleife" drehen. Im obigen Beispiel ist das die Seite „b2" = Wurzel aus($a^2+c^2-2ac*\cos\beta_2$). Zum Thema Kongruenzsätze wird hier nichts erwähnt. Diese sind für die Erstellung der Software uninteressant.

Beispiel einer Dreiecksberechnung mit zwei Lösungen

Gegeben: a = 3, b = 4 und α = 45°

switch case	a	b	c	α	β	γ	Konstruktion	
2	●	●		●			SSW	

Zuerst wird der Winkel Beta (β) mit dem Sinussatz berechnet:

$$\frac{\sin\beta}{b} = \frac{\sin\alpha}{a} \Rightarrow \sin\beta = \frac{\sin\alpha \;\; b}{a} = \frac{4\sin(45)}{3} = \underline{70{,}53°}$$

Für den Winkel Beta (β) gibt es nun zwei Möglichkeiten:

Winkel β_1 = $\underline{70{,}53°}$ Winkel β_2 = 180° - 70,53° = $\underline{109{,}47°}$ *1

Mit Hilfe der Winkelsumme im Dreieck werden die möglichen Werte für Gamma (**γ**) berechnet:

Winkel γ_1 = 180° - 45°- 70,53° = $\underline{64{,}47°}$ Winkel γ_2 = 180° - 45°- 109,47° = $\underline{25{,}53°}$ *1

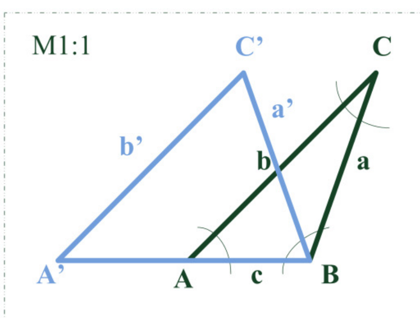
Anschließend wird mit dem Cosinussatz die Seite c berechnet:

$c^2 = a^2+b^2-2ab\cos\gamma$ $\quad c_1 = \sqrt{3^2+4^2-2*3*4*\cos(64{,}47)} = \underline{3{,}83}$

$$c_2 = \sqrt{3^2+4^2-2*3*4*\cos(25{,}53)} = \underline{1{,}83}$$

Damit haben wir zwei gültige Lösungen:

β_1 = 70,53	**β_2 = 109,47**
γ_1 = 64,47	**γ_2 = 25,53**
c_1 = 3,83	**c_2 = 1,83**



*1 Bitte beachten: Sind die berechneten Winkel negativ, dann gibt es keine zweite Lösung. In der Software wird im obigen Fall „**β_2 ist negativ**" die weitere Berechnung abgebrochen.

Noch ein paar Worte zu den vorherigen Seiten. Das Programm „DasDreieck“ muss für die Dreiecksberechnung 28x3 = 84 Lösungswege bereitstellen. In der Programmierung können wir nicht 84 Algorithmen für die Visualisierung des Dreiecks einschließlich des Lösungswegs herunterleiern. Wir müssen Gemeinsamkeiten finden und Methoden schreiben. Und was die Programmierer gerne tun, ist Interfaces zur Verfügung zu stellen, dann kann ein Grafiker daraus Visualisierungsdateien erstellen. Genau das tu ich auch. Nur das ich die Interfaces und die Grafikdateien selber bauen muss. In einem Verzeichnis: “C:\WorkingDirectory\Resource\Bitmaps” sind Grafikdateien hinterlegt, zum Beispiel “triangle_1.png”. Diese werden von der Software aufgerufen und in Abhängigkeit der XY-Pixel-Positionen werden Zahlenwerte eingefügt.
Was läuft da nun genau ab? Der Anwender hat sechs Eingabemöglichkeiten. Davon muss er drei Werte editieren. Schon während der Editierung werden die Eingabedaten auf Plausibilität geprüft und es wird eine Datenstruktur „Dreieck" gefüllt, siehe Bild. Hinter jedem Eingabefeld wird eine Kennung hinterlegt. Für Dreiecks-Strecken „abc“ für Dreiecks-Winkel „xyz“. Egal in welcher Reihenfolge die Daten eingegeben werden, ein Sortieralgorithmus (in der C++-Sprache ist das ein „Ein-Zeiler“) baut daraus eine „Drei-Aus-Sechs-Funktion“ Diese Funktion gibt einen Zahlenwert von 1 bis 19 zurück und füttert damit eine SWITCH/CASE-Anweisung, die zwangsläufig 19 Verzweigungen besitzt. Eigentlich müssten es ja 20 Werte sein, aber eine Dreiecksberechnung nur aus Winkel ist unsinnig und wird ausgeblendet. Als nächstes wird die Anzahl von 16 Formeln auf insgesamt fünf Funktionen gebracht.

```
//**************************************************************************
//      Applikation Dreieck, Struktur des Dreiecks
//      QT-Creator:3.5.1 Based on Qt 5.5.1. Datum: 14.04.2016
//                       (c) Hans Busche
//**************************************************************************

#include <QtGlobal>
#include <QString>
#include <QStringList>

#ifndef STRUCT_TRIANGLE
#define STRUCT_TRIANGLE

struct dreieck
{
    //1.Lösung:
    qreal a;                //Seite a
    qreal b;                //Seite b
    qreal c;                //Seite c
    qreal alpha;            //Winkel alpha
    qreal beta;             //Winkel beta
    qreal gama;             //Winkel gamma
    bool istRechtwinklig;   //ist das Dreieck rechtwinklig?
    int step;               //case-Anweisung (switch)
    int anzWerte;           //Zähler für "gegebene" Werte (rechtw.Dreieck:2)
    QStringList s;          //Kennung "gegeben"
    QString kongruenz;      //Kongruenzsätze
    int error;              //Fehlermeldung oder keine Lösung
    int zweiLoesung;        //2. Lösung vorhanden
    //2.Lösung
    qreal a2;               //Seite a
    qreal b2;               //Seite b
    qreal c2;               //Seite c
    qreal alpha2;           //Winkel alpha
    qreal beta2;            //Winkel beta
    qreal gama2;            //Winkel gamma
};

#endif // STRUCT_TRIANGLE
```

16 benötigte Formeln werden auf 5 Software-Funktionen reduziert

F Nr	nichtrechtwinklig	Switch/Case Software	rechtwinklig	Switch/Case Software
1	$a = \frac{\sin\alpha * c}{\sin\gamma}$	17;18,19		
2	$a = \frac{\sin\alpha * b}{\sin\beta}$	14;15,16	$a = c * \sin\alpha$	5;6;7;8;9
3	$a^2=b^2+c^2-2bc*\cos\alpha$	11;12,13		
4	$b= \frac{\sin\beta * a}{\sin\alpha}$	8;9;10;17;18;19	$b = c * \sin\beta$	8;9;2;3;4
5	$b^2=a^2+c^2-2ac*\cos\beta$	6;5;7		
6	$c= \frac{\sin\gamma * a}{\sin\alpha}$	8;9;10;14;15;16	$c = \frac{a}{\sin\alpha}$	1;3;4
7	$c^2=a^2+b^2-2ab*\cos\gamma$	4;2;3	$c = \frac{b}{\sin\beta}$	6;7
8	$\alpha = 180°-\beta-\gamma$	10;16;19;12;13	$\alpha = 180°-\beta-\gamma$	4;7;9;5
9	$\sin\alpha= \frac{\sin\beta * a}{b}$	3;7	$\sin\alpha=\frac{a}{c}$	2
10	$\cos\alpha= \frac{b^2+c^2-a^2}{2bc}$	1;4;6		
11	$\beta = 180°-\alpha-\gamma$	9;15;18;5;7	$\beta = 180°-\alpha-\gamma$	3;6;8;2;1
12	$\sin\beta=\frac{\sin\alpha * b}{a}$	2;13	$\sin\beta=\frac{b}{c}$	5
13	$\cos\beta=\frac{a^2+c^2-b^2}{2ac}$	1;11;4	$\tan\alpha =\frac{a}{b}$	1
14	$\gamma = 180°-\alpha-\beta$	8;14;17;2;3		
15	$\sin\gamma=\frac{\sin\alpha * c}{a}$	5;12	$\gamma = 90°$	
16	$\cos\gamma=\frac{a^2+b^2-c^2}{2ab}$	1;6;11		

Für die Berechnung des nichtrechtwinkligen Dreiecks werden 16 Formeln benötigt. In der Software werden diese Formeln auf insgesamt 5 Funktionen reduziert. Nachfolgend ein Beispiel:

Funktion -1- für Formeln: 9,12,13 und 15

siehe „FNr“ oben links

```
qreal mathdreieck::SinusSatzComputingAngle
              (qreal Multiplikator, qreal Dividend, qreal Winkel)
{
   //sin alpha = sin(beta) * a / b
   //sin beta  = sin(alpha) * b / a
   //sin gamma = sin(alpha) * c / a
   qreal v,w;
   w = qSin(Winkel*PI/180.0)*Multiplikator/Dividend;
   v = qAsin(w) * 180.0/PI;
  //Plausibilitätsprüfung
   if(IsFloatError(v) == false)
      return(v);
   return(0);
}
```

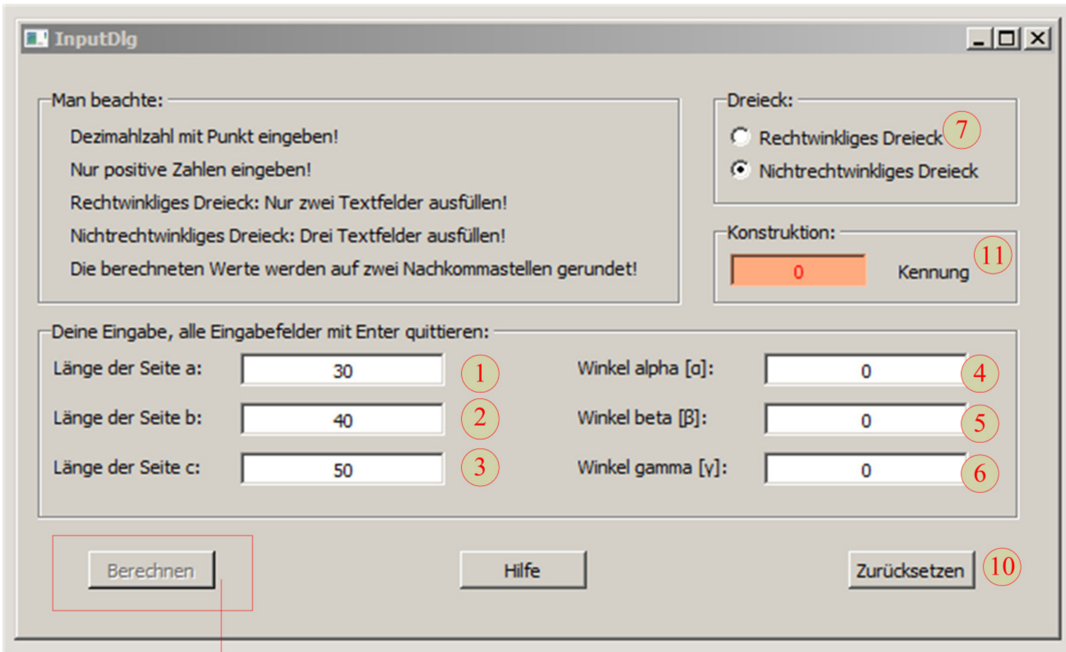
Funktion -2- Sinussatz Seite	Für Formeln 1,2,4,6 [FNr]
Funktion -3- Cosinussatz Seite	Für Formeln 10,13,16 [FNr]
Funktion -4- Cosinussatz Winkel	Für Formeln 1,2,4,6 [FNr]
Funktion -5- Winkeldifferenz	Für Formeln 8,11,14 [FNr]

Auf der vorherigen Seite, unten links, sieht man die C++ Funktion für die Berechnung des Winkels mit dem Kosinus Satz. Das Qt-Framework sowie fast alle C++ Bibliotheken verwenden in der Trigonometrie das Bogenmaß. Hier kurz der Zusammenhang zwischen Gradmaß und Bogenmaß: arc 360° = 2π = 6,28319 -> arc 1° ist 2π/360 = π/180 = 0,01745. Der Sinus von Eins ist 90°. Der Kosinus von Eins ist Null. Was passiert, wenn du mit deinem Taschenrechner einen Sinus von 1,2 bzw. Kosinus von 1,2 umrechnen willst? Das Display des Taschenrechners zeigt „Error“ an. Leider generiert das Qt-C++-Framework keine Ausnahme (Exception) für die oben beschriebene Situation. Ein „try{} / catch(..)-Block reagiert hier nicht auf „Infinity-Unendlich“ entsprechend. Die Berechnung wird bis zum Schluss durchgeführt und in den Float-Variablen steht irgendetwas Unsinniges drin. Jedoch wenn es für ein Dreieck keine Lösung gibt, dann entsteht zwangsläufig so eine Situation. Deshalb muss der Programmierer hier selbst einen Abfangjäger bauen, um zu erkennen, dass eine Ausnahme aufgetreten ist. Ich habe mich für eine String-Methode entschieden, siehe Bild:

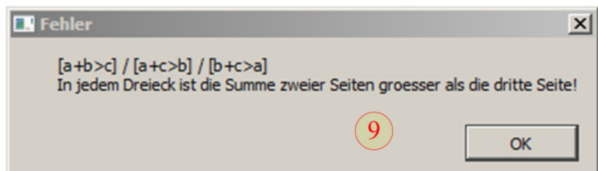
```
//Exception, wenn ein Sinus- Kosinus Wert ausserhalb
//des Wertebereiches liegt. ODER der Winkel negativ ist.
bool mathdreieck::IsFloatError(qreal value)
{
    QString s;
    s = s.number(value,'f',2);
    //Erkennung Float-Infinity mittels
    //String-Konvertierung! Excetion:"nan"
    if((s == "nan")||(value < 0))
    {
        trDat->error += -1;
        return(true);
    }
    return(false);
}
```

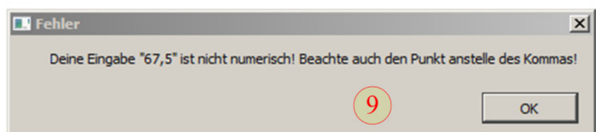
Später hat sich dann herausgestellt, dass ich die System-Portierbarkeit verliere. Ein Alleinstellungsmerkmal des Qt-C++-Frameworks ist die hundertprozentige Portierbarkeit. Ja! Das ist ein schweres Vergehen unter Programmierern. Unter Windows wird der Name „nan“ erzeugt, jedoch nicht unter Linux. Wie ich damit umgehe weiß ich noch nicht. Auf jedem Fall muss ich da nochmal ran. Gott sei Dank! Die Reparatur erfolgt in einer Klasse und wenn die stimmt, dann sind alle anderen Programmteile automatisch auch in Ordnung.
Weitere Details zu diesem Thema:
http://www.gnu.org/software/libc/manual/html_node/Infinity-and-NaN.html

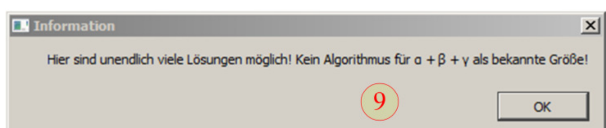
Das Eingabe-Dialogfeld



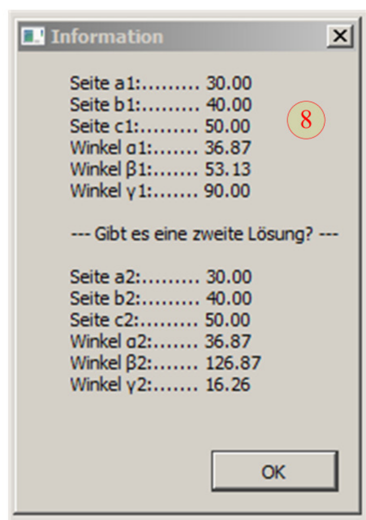
Der Schalter „Berechnen“ wird freigeschaltet wenn: In Abhängigkeit des Dreiecktyps (7) die Daten in die Eingabefelder (1 bis 6) fehlerfrei eingegeben worden sind. Beim rechtwinkeligen Dreieck sind das 2 Eingaben von sechs und beim nichtrechtwinkligen Dreieck sind das 3 Eingaben von sechs. Bei Fehleingaben (siehe rechts) werden alle vorherigen Zuweisungen verworfen. Beachte, dass auch der Typ des Dreiecks dann wechselt (7).



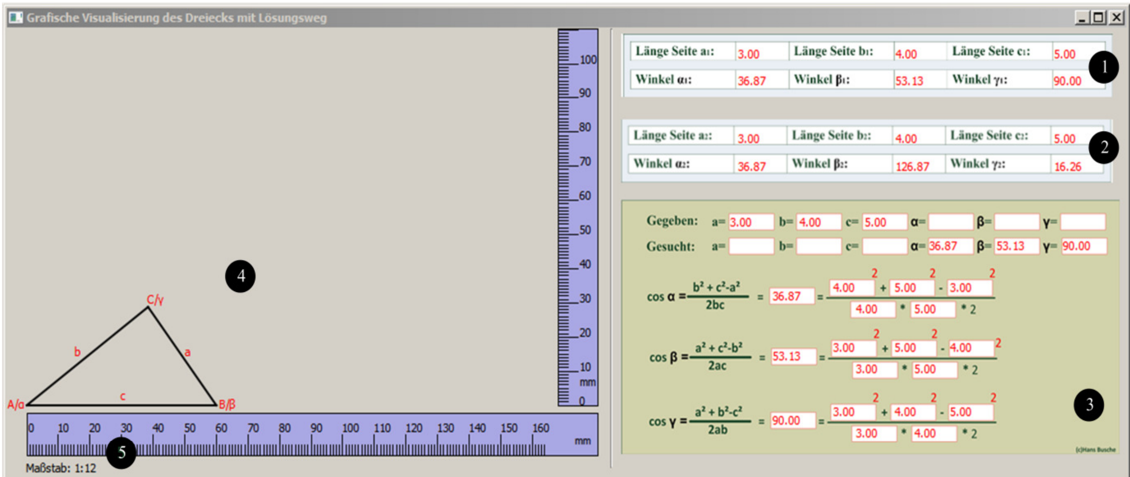




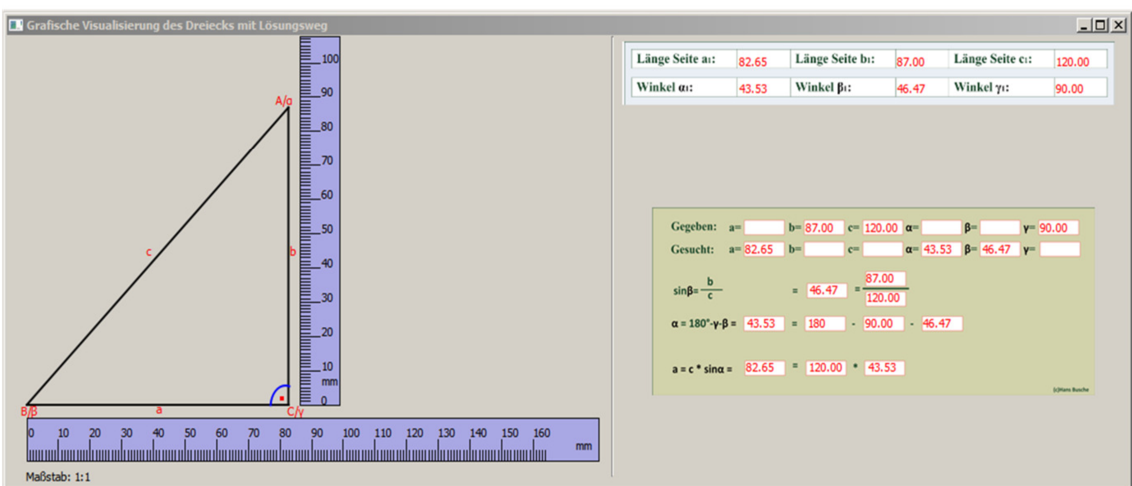
Bist du an der Visualisierung des Dreiecks einschließlich Lösungsweg nicht interessiert, sondern möchtest schnell nur die Lösung, so gibt es das auch (8). Für die Plausibilitätsprüfung gibt es insgesamt 7 Fehler- bzw. Informationsmeldungen (9). Möchtest du mehrere Berechnungen durchführen, dann muss vor dem Beginn der nächsten Berechnung die Taste „Zurücksetzen“ (10) einmal betätigt werden. Position (11) Konstruktion / Kennung ist ein Informationsfeld. Eingaben werden dort nicht getätigt. Wenn der Anwender für die Dreiecksberechnung die Seiten a,b und c editiert, dann wird im Feld (11) die Information „sss/1“ eingetragen. Bedeutet: Seite/Seite/Seite und switch/case – Verzweigung 1, siehe auch Kapitel „Formel-Matrix“ weiter oben in dieser Dokumentation.



Besonderheiten der grafischen Darstellung

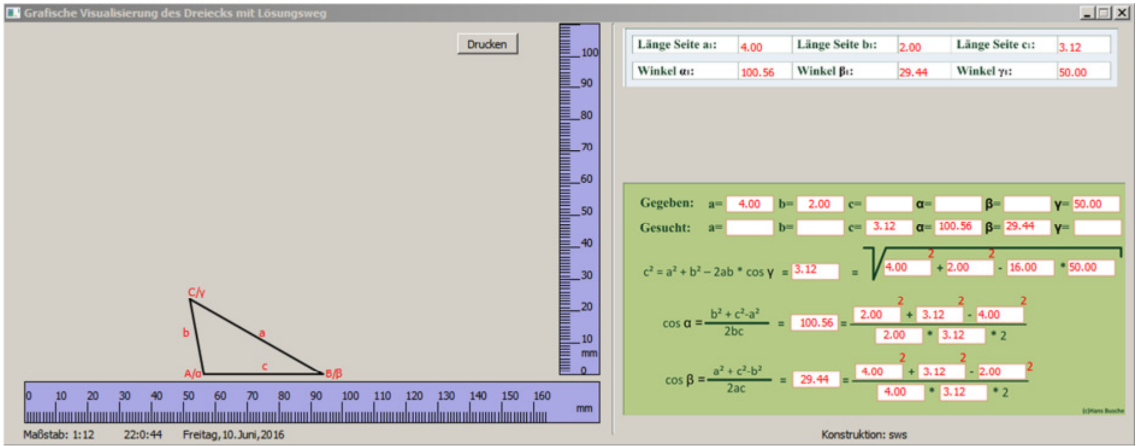


[1 und 2] Das Dreieck mit den gegebenen Seiten a=3, b=4, c=5 hat zwei Lösungen. [3] Lösungsweg. [4] Leinwand für die grafische Darstellung des Dreiecks. [5] Skalierung (Maßstab) für das Dreieck. Die Skalierung geht in zwei Richtungen: Vergrößerung und Verkleinerung. Im obigen Fall kann das Dreieck im Maßstab 1:1 nicht dargestellt werden, deshalb wurde es auf das 12fache vergrößert. Sind Dreiecksseiten größer 160/100mm dann wechseln die blauen Lineare und die Darstellung erfolgt in Pixel. Sind die Dreiecksseiten außerhalb des Anzeigebereiches (600x400 Pixel) dann bleibt die Zeichenfläche leer. Es wird ein Hinweistext eingeblendet.



Darstellung eines rechtwinkligen Dreiecks. Selbstverständlich nur eine Lösung. Sinusfunktion als Lösungsweg. Vertikale Lineal folgt der Seite b. Die Lineale haben noch einen Fehler. Die Millimeter-Auflösung stimmt nicht. Irgendetwas habe ich mit den Themen „*Pixel / DPI / Millimeter*“ noch nicht ganz verstanden. Maßstab 1:1.

Auf eine Einschränkung muss ich noch hinweisen: Die nachfolgende Grafik zeigt ein Dreieck, dessen Winkel α ein stumpfer ist (100,6°) Dadurch kann es nicht am linken Rand ausgerichtet werden. Es verschiebt sich waagerecht zur Mitte. Das waagerechte Lineal müsste eigentlich auch neu ausgerichtet werden. Na ja, das wird wohl so bleiben.



Berechnetes Dreieck ausdrucken

Pixel, DPI, PPI und Auflösung. Von Pixel nach Millimeter und umgekehrt.

Dieses Kapitel habe ich parallel zur Programmierung geschrieben. Ich nehme es in dieser Dokumentation mit auf, weil ich einige Fehler gemacht habe. Foren im Internet. Manchmal ist das ein Kreuz. Deutsche Foren sind oft voller Eitelkeiten und Fehlinformationen. Meine Erfahrung: Quäle dich durch englischsprachige Foren, sie sind technikfreudig und beleidigen nicht.

Pixel: Picture Element = Bildpunkte. Kleinste Einheit eines Bildes. Jeder Pixel steht für eine bestimmte Farbinformation. Diese ergibt sich aus der Mischung der Farben Grün, Blau und Rot. Somit belegt ein Pixel 3 Byte Speicherplatz. Pixel können unterschiedlich groß sein.

PPI: Pixel per Inch = Pixel pro 2,54 cm. 72x72 PPI bedeutet auf einer Fläche von 2,54cm x 2,54cm befinden sich 5184 Pixel.

DPI: Dots per Inch = Druckpunkte pro Inch. Unter “Dots” versteht man hier die Punkte bzw. Farbtröpfchen, die ein Drucker aufs Papier bringt. Ansonsten gleiche Definition wie PPI. Ein digitales Bild auf dem Computer hat niemals DPI, cm oder Inch als Größenangabe sondern nur Pixel.

Merksatz: Solange ein digitales Foto nicht gedruckt werden soll, ist die PPI/DPI-Angabe völlig bedeutungslos.

Datenkompression: Wird ein Digitalbild nicht-komprimiert gespeichert braucht es bei einer Auflösung von 1200x800 Pixel 2,88Mbyte Speicherplatz (1200*800*3). Deshalb gibt es komprimierte Dateiformate wie JPG.

Merksatz: Ab jetzt schmeiß ich die Begriffe PPI und DPI in einen Topf und verwende nur noch DPI.

Auflösung: Computer / Bildschirm in Pixel. Zum Beispiel 1920 x 1080 Pixel. Drucker und Scanner (Abtastpunkt wie Druckpunkt) in DPI.

Der mathematischen Zusammenhang: **Px = Pixel,   D = DPI, L = Zentimeter (cm)**.

$$\mathbf{Px} = \frac{\boldsymbol{L * D}}{\mathbf{2,54}}$$

$$\mathbf{L} = \frac{\boldsymbol{Px} * \mathbf{2,54}}{\boldsymbol{D}}$$

$$\mathbf{D} = \frac{\boldsymbol{Px} * \mathbf{2,54}}{\boldsymbol{L}}$$

Beispiel 1: Eine Dreiecksfigur soll auf ein DIN-A4-Blatt ausgedruckt werden. Damit eine Seite des Dreiecks mit anliegendem Winkel von 37° nicht aussieht wie ein Sägezahn muss mit 300 DPI gedruckt werden. Was für eine Pixel-Auflösung ist erforderlich? DIN A4 Format = 21,0cm x 29,7cm. Pixel Breite = 21,0cm * 300 DPI / 2,54 = 2480 Pixel. Pixel Länge = 29,7cm * 300 DPI / 2,54 = 3508 Pixel. Der Computer muss dem Drucker einen Datensatz von *2480x3508 Pixel (26 Mbyte)* schicken.
Beispiel 2: Ein Digitalbild, 5000x3500 Pixel groß, soll auf ein DIN-A3-Blatt gedruckt werden (42,0cm x 29,7cm) Die Druckauflösung soll mindestens 300 DPI betragen. Ist das Bild verwendbar? 42,0cm*300DPI/2,54cm=4960 Pixel. Breite ist in Ordnung. 29,7cm*300DPI/2.54cm=3508 Pixel. In der Länge wird es aber knapp.

Die Seiten des Dreiecks in Millimeter zeichnen.

In der Programmiersprache C++ werden Bibliotheken für Grafik-Implementierung bereitgestellt. Will man in Abhängigkeit eines Winkels eine Linie zeichnen, so gibt es viele Möglichkeiten, hier eine davon:

```
QLineF a,b,c;
a.setP1(QPointF(x,y));
a.setAngle(w);
a.setLength(z);
```

Die Seite “a” wird vom Startpunkt „xy“ mit einem Winkel von “w” und einer Länge von „z“ in ***Pixel*** gezeichnet. Jedoch soll die Seitenlänge hier in Millimeter dargestellt werden und nicht in Pixel. Um 43mm zu zeichnen, wie viele Pixel müssen unter „z“ eingetragen werden?

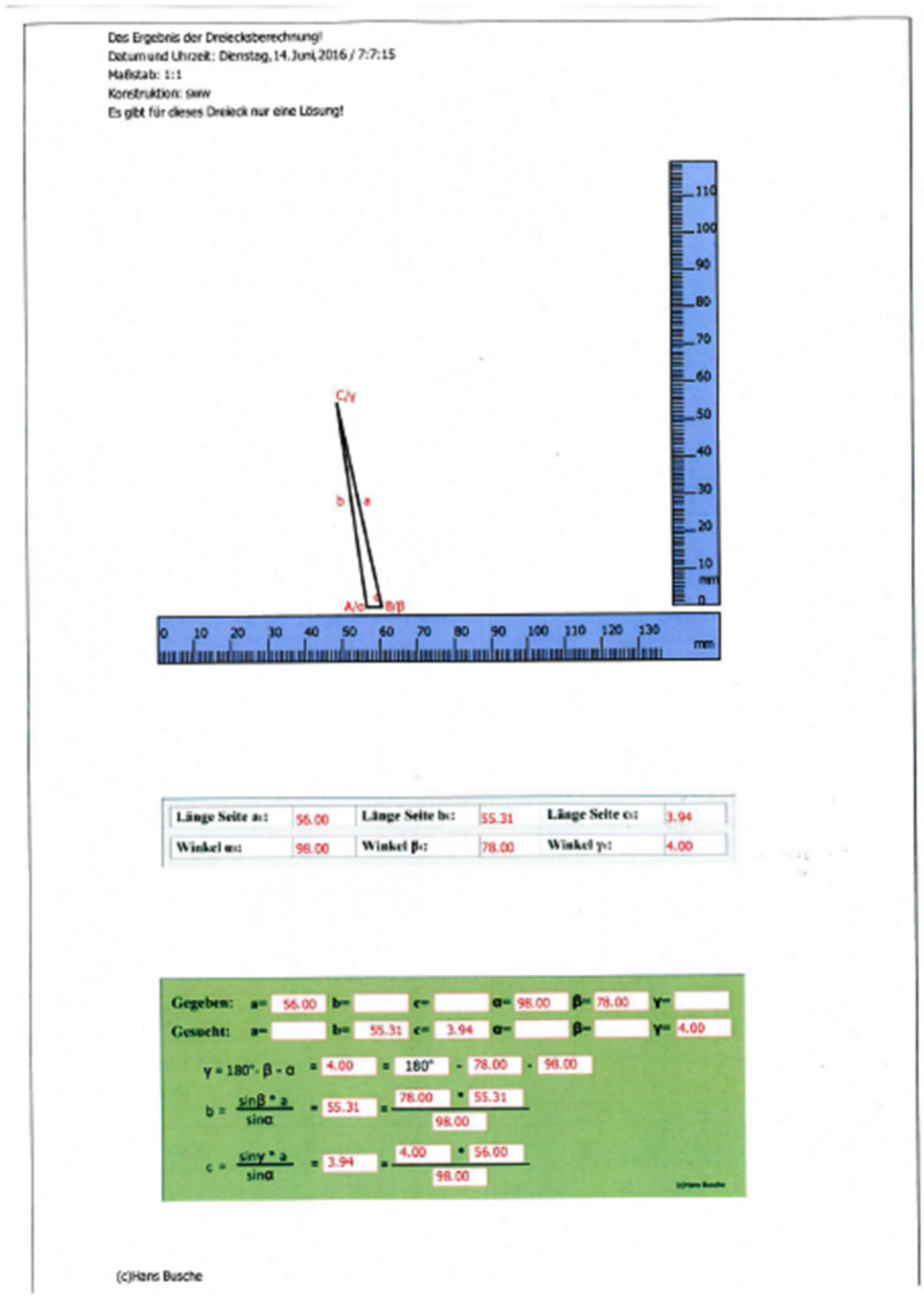
Bevor wir diese Frage klären, hier noch einen Hinweis: Die C++-Klasse „triangleGraph“ bekommt einen Parameter DPI übergeben. Dieser DPI-Wert wird für den Bildschirm mit 81 DPI und für den Drucker mit 300 DPI belegt. Im Gegensatz zur Aussage „DPI nur für Drucker“ auf der vorherigen Seite! DPI-Drucker ist klar, aber DPI für den Bildschirm! Warum? Sehen wir gleich!
Mein 27“-Bildschirm hat eine Auflösung von 1920x1080 Pixel. Weiterhin 60,0cm x 33,5cm Sichtfläche. DPI=1920*2,54 / 60,0cm = 81.

Millimeter pro Pixel $= \frac{\mathbf{25,4}}{\boldsymbol{DPI}}$ **= 25,4/81= 0,3 Millimeter/Px für den Bildschirm**

Millimeter pro Pixel $= \frac{\mathbf{25,4}}{\boldsymbol{DPI}}$ **= 25,4/300= 0,08 Millimeter/Px für den Drucker**

Für die Funktion ***„a.setLength(z)“*** die dafür sorgt, dass eine Linie mit „z“ Pixel gezeichnet wird, muss bei einer gewünschten Länge von 43mm einen z-Wert von 43/0,3=143 Pixel für den Bildschirm und 43/0,08= 537 Pixel für den Drucker bekommen.

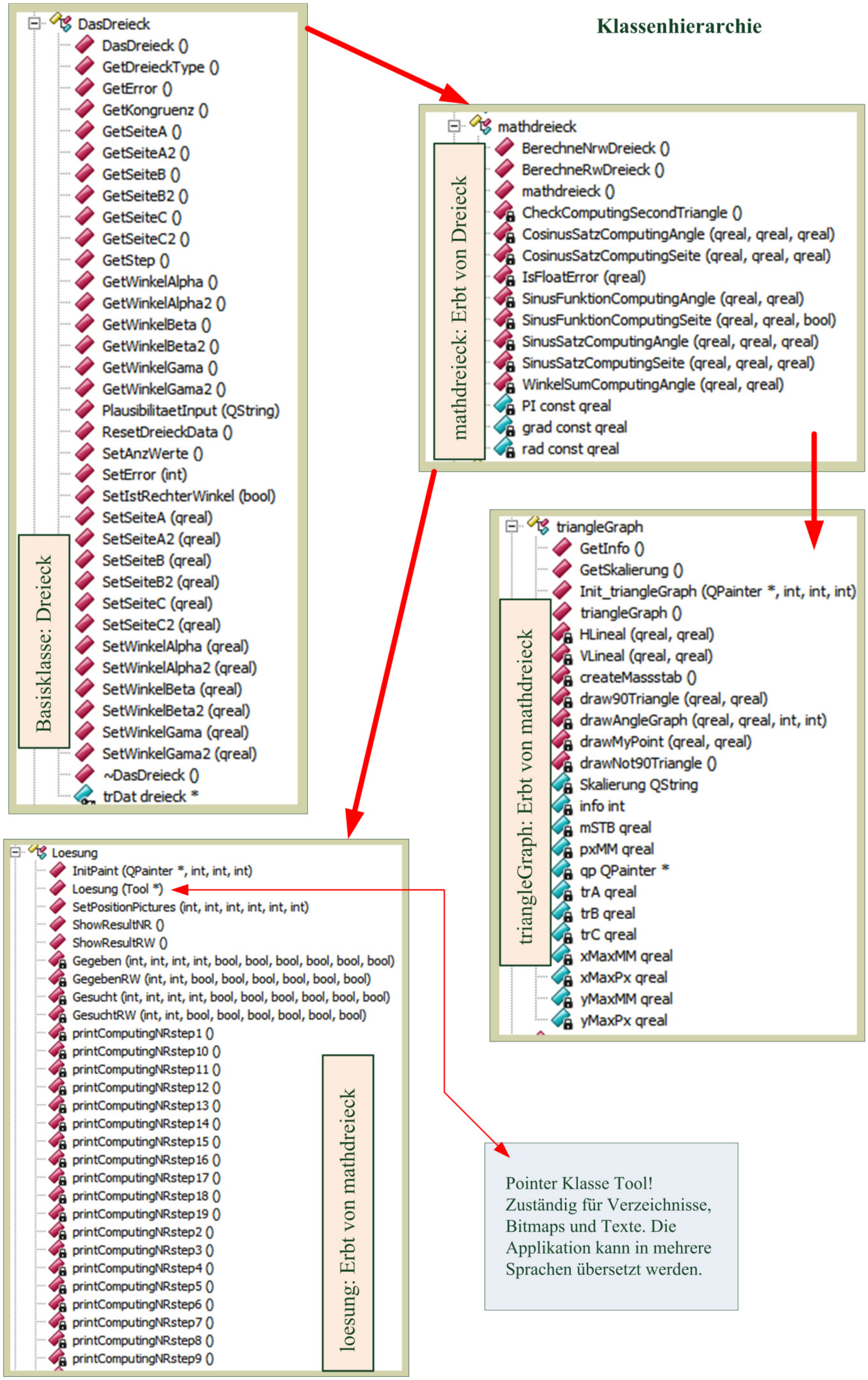
Auf der nachfolgenden Seite sieht man einen Ausdruck einer Dreiecksberechnung.



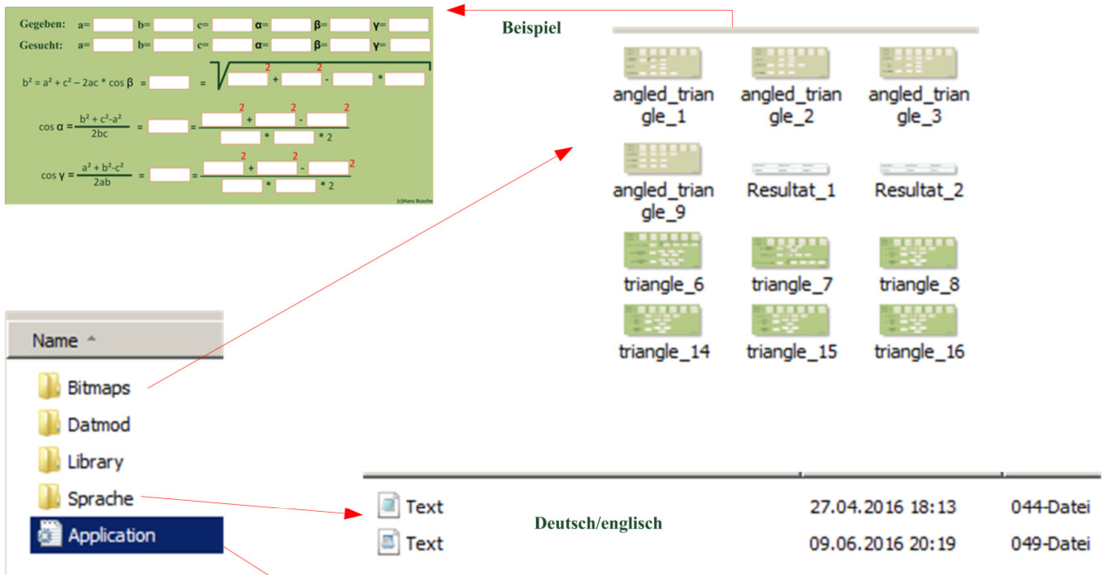
Beispiel DIN-A4-Printout

Grenzfall einer Dreiecks-Visualisierung. Die automatische Positionierung der Bezeichner stoßen an ihre Grenzen Die Beschriftung der Seite „c“ wird von der Seitenlinie „a“ verdeckt.

Klassenhierarchie



Verzeichnisse, Bitmaps, Texte und Konfiguration, eine Aufgabe der Klasse „Tools“



Beispiel: Programmteil aus der Klasse „Tool“

```
/**
 * @brief Tool::IstEsEinSchaltJahr.\n
 * @param unsigned short Jahr
 * @return bool, true=ein Schaltjahr
 */
bool Tool::IstEsEinSchaltJahr(unsigned short Jahr)
{
    if((Jahr%4 == 0) && (Jahr%100 != 0) || (Jahr%400 == 0))
        return(true);
    return(false);
}
```

Application.ini

```
[General]
Copyright=(c)Hans Busche
Opensource=Qt-Creator 3.5.1
Framework=Qt5.5.1
CurrentCpuArchitecture=x86_64
KernelType=winnt
PrettyProductName=Windows 7
AppEnde=12.11.2015 16:58:47
AppStart=12.11.2015 16:58:47

[RESOURCES]
BitmapFolder=/Resource/Bitmaps
DatmodFolder=/Resource/Datmod
LanguageFolder=/Resource/Sprache
LibraryFolder=/Resource/Library

[LANGUAGES]
Language=049
english=044
deutsch=049
```

Bevor du irgendeine Anwendung programmierst, solltest du dir Gedanken machen, wie du deine Software organisierst. Da gibt es viele Möglichkeiten. Bei allen meinen Programmen schleppe ich die Klasse "Tool" mit. Sie übernimmt folgende Aufgaben:

1) Globaler String zum Arbeitsverzeichnis.
2) Anlegen der Verzeichnisse für: Daten, Bitmaps, Sprachen und Bibliotheken.
3) Konfiguration, Protokolle.
4) Methoden für Datum und Uhrzeit, Exception, Vektoren für Texte, Vektoren/Listen für temporäre Daten usw.
5) Lesen und Schreiben von Dateien.
6) Abfrage Betriebssystem (Windows / Linux usw.)

Die Konfigurationsdatei „Applikation.ini“ organisiert die Programmumgebung. Weiterhin führt sie auch ein Protokoll.

Text, Sprache

Wenn dir die Texte nicht gefallen so kannst du diese ändern. Beim Hochfahren der Software werden die Texte in einen Hashtable geladen. Ein benötigter Text wird mit der Klassen-Methode *„public GetText(offset)“* angezeigt. Offset ist die Adresse des Textes z. B. ein Offset von 12 zeigt den Text "Mai" an. Das bedeutet die Adresse "12=" darf nicht geändert werden. Mit einem Texteditor (ich benutze "Notepad++") kannst du die Texte ändern. Erstelle aber vorher eine Sicherungskopie. Der Pfad: \Resource\Sprache

```
7=Sonntag
8=Januar
9=Februar
10=März
11=April
12=Mai
13=Juni
14=Juli
15=August
16=September
17=Oktober
18=November
19=Dezember
20=Qt-Creator 3.5.1
21=Qt5.5.1
22=(c)Hans Busche
;---> kann geändert werden
23=Fehler
24=Deine Eingabe
25=ist nicht numerisch! Beachte auch den Punkt anstelle des Kommas!
26=Bei einem rechtwinkligem Dreieck kann ein zweiter/dritter Winkel nicht 90° oder groesser sein!
27=Deine Eingabe für das Rechtwinklige Dreieck: [a ist groesser c] wird nicht akzeptiert!
28=Deine Eingabe für das Rechtwinklige Dreieck: [b ist groesser c] wird nicht akzeptiert!
```

Software

Das Programm „DasDreieck“ wird nicht verkauft. Es wird verschenkt. Jeder kann es nutzen. Garantie- und/oder Gewährleistung ausgeschlossen. Jeder ist aufgefordert den Datenträger, der alle Dateien enthält, damit die Software läuft, mit einem Virenscanner zu überprüfen.

Besonderheiten Projektverwaltung Qt-C++-Creator

Compiler Anweisungen und explizite Zuweisung Bibliotheken

```
#-------------------------------------------------
#
# Project created by QtCreator 2016-06-04T21:16:06
#
#-------------------------------------------------

QT       += core gui
QT       += printsupport

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = Triangle
TEMPLATE = app


SOURCES += main.cpp\
        inputdlg.cpp \
    dasdreieck.cpp \
    loesung.cpp \
    mathdreieck.cpp \
    tools.cpp \
    trianglegraph.cpp \
    graphtriangle.cpp \
    printtriangle.cpp

HEADERS  += inputdlg.h \
    dasdreieck.h \
    loesung.h \
    mathdreieck.h \
    struct_triangle.h \
    tools.h \
    trianglegraph.h \
    graphtriangle.h \
    printtriangle.h

FORMS    += inputdlg.ui \
    graphtriangle.ui

```

Erstes linkes Bild: Einstellung für den Microsoft Visual C++ Compiler (MSVC12).
Achtung: Alles wird automatisch generiert, bis auf die Anweisung:
QT += printersupport
Diese muss manuell eingefügt werden.

```
#-------------------------------------------------
#
# Project created by QtCreator 2016-06-04T21:16:06
#
#-------------------------------------------------

QT       += core gui
QT       += printsupport
CONFIG   += c++11

greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = Triangle
TEMPLATE = app


SOURCES += main.cpp\
        inputdlg.cpp \
    dasdreieck.cpp \
    loesung.cpp \
    mathdreieck.cpp \
    tools.cpp \
    trianglegraph.cpp \
    graphtriangle.cpp \
    printtriangle.cpp

HEADERS  += inputdlg.h \
    dasdreieck.h \
    loesung.h \
    mathdreieck.h \
    struct_triangle.h \
    tools.h \
    trianglegraph.h \
    graphtriangle.h \
    printtriangle.h

FORMS    += inputdlg.ui \
    graphtriangle.ui

```

Zweites linkes Bild: Einstellung für MinGw 4.9.2-32Bit Compiler.
Achtung: Alles wird automatisch generiert, bis auf die Anweisungen:
QT += printersupport
CONFIG += c++11
Diese müssen manuell eingefügt werden.