

# Besondere Lernleistung

Formula



## Inhaltsverzeichnis

---

Kapitel	Titel	Seiten
1	<a href="#">Einleitung</a>	5
2	<a href="#">Wie stellt man eine mathematische Formel auf dem Computer dar?</a>	6
1	<a href="#">Das Formelobjekt</a>	7
2	<a href="#">Der Titel einer Formel</a>	7
3	<a href="#">Der Körper einer Formel</a>	7
4	<a href="#">Die Informationsdaten einer Formel</a>	8
3	<a href="#">Wie speichert man solche Formeln dauerhaft?</a>	8
1	<a href="#">Das TFile-Objekt</a>	8
2	<a href="#">Das Dictionary</a>	9
3	<a href="#">Das Speichern einer Formel</a>	10
4	<a href="#">Wie findet man Lösungen für Problemstellungen?</a>	10
1	<a href="#">Formeln finden</a>	10
2	<a href="#">Der Lösungsbaum</a>	12
3	<a href="#">Rekursion</a>	13
4	<a href="#">Die Formelsequenz</a>	13
5	<a href="#">Wie rechnet man Formeln mit Hilfe mathematischer Funktionen aus?</a>	14
1	<a href="#">Das Modell des Parsing</a>	16
2	<a href="#">Der Parserstring</a>	16
3	<a href="#">Das eigentliche Parsing</a>	17
4	<a href="#">Das Ausrechnen verketteter Formeln</a>	17
6	<a href="#">Wie stellt man die mathematischen Grundfunktionen numerisch stabil dar?</a>	18
1	<a href="#">Das Abfangen von Exceptions</a>	18
2	<a href="#">Die erweiterten Zustände – der Datentyp tRItem</a>	18
7	<a href="#">Wie schafft man es, alle möglichen Lösungen zu beachten?</a>	19
1	<a href="#">Verwalten mehrerer Lösungen</a>	20
2	<a href="#">Rechnen mit mehreren Lösungen</a>	20
3	<a href="#">Ausrechnen</a>	22
8	<a href="#">Wie erstellt man eine sinnvolle, möglichst selbsterklärende, Bedienoberfläche?</a>	23
9	<a href="#">Rückblick und Ausblick</a>	25
	<a href="#">Anhänge</a>	27
A	<a href="#">Beispiele für die Arbeit mit Formula</a>	29
B	<a href="#">Die Hilfedatei von Formula</a>	33
C	<a href="#">Der Quelltext der aktuellen Version des praktischen Teils</a>	45
1	<a href="#">Die Unit BaseBox.pas</a>	45
2	<a href="#">Die Unit Common.pas</a>	48
3	<a href="#">Die Unit Constants.pas</a>	60
4	<a href="#">Die Unit Constgrid.pas</a>	65
5	<a href="#">Die Unit Dictionary.pas</a>	67
6	<a href="#">Die Unit Errors.pas</a>	77
7	<a href="#">Die Unit Files.pas</a>	81

8	<a href="#">Die Unit Formelbase.pas</a>	88
9	<a href="#">Die Unit Formelgrid.pas</a>	96
10	<a href="#">Die Unit Formels.pas</a>	106
11	<a href="#">Die Delphi-Projektdatei Formula.dpr</a>	114
12	<a href="#">Das Delhipackage Formula.dpk</a>	115
13	<a href="#">Die Unit GridRoot.pas</a>	116
14	<a href="#">Die Unit Ladenpage.pas</a>	123
15	<a href="#">Die Unit Lineview.pas</a>	125
16	<a href="#">Die Unit Mainform.pas</a>	136
17	<a href="#">Die Delphi-Formulardatei Mainform.dfm</a>	143
18	<a href="#">Die Unit Parser.pas</a>	149
19	<a href="#">Die Unit Parseritem.pas</a>	155
20	<a href="#">Die Unit Reell.pas</a>	161
21	<a href="#">Die Unit ReellErrors.pas</a>	200
22	<a href="#">Die Unit Ritems.pas</a>	207
23	<a href="#">Die Unit Rtype.pas</a>	218
24	<a href="#">Die Unit SolutionView.pas</a>	228
25	<a href="#">Die Unit xReell.pas</a>	242
<u>D</u>	<a href="#">Quellcode der ersten ausführbaren Version von Formula</a>	285
1	<a href="#">Die Unit Database.pas</a>	285
2	<a href="#">Die Programmdatei Dataread.pas</a>	304
3	<a href="#">Die Programmdatei Formel.pas</a>	306
4	<a href="#">Die Unit Formula.pas</a>	309
5	<a href="#">Die Unit FuncBase.pas</a>	316
6	<a href="#">Die Unit Objects.pas</a>	319
7	<a href="#">Die Unit Pas.pas</a>	338
8	<a href="#">Die Unit Reel.pas</a>	344
<u>E</u>	<a href="#">Protokoll</a>	377
<u>F</u>	<a href="#">Quellenverzeichnis</a>	379
<u>G</u>	<a href="#">Danksagung</a>	381

Der theoretische Teil meiner Besonderen Lernleistung entstand erst nach Beendigung der praktischen Arbeit und soll diese in chronologischer Reihenfolge nachvollziehen. Dabei bezieht er sich auf die letzte Version von Formula.

Weil es sich um ein Programm handelt, ist es unumgänglich Fachwörter aus dem Bereich der Informatik zu verwenden. Diese werden jedoch stets erklärt. Ebenso werden Zusammenhänge in beispielsweise Quelltextauszügen durch farbliches Hervorheben verdeutlicht.

Da der Umfang des theoretischen Teils auf nur zwanzig Seiten beschränkt ist, kann ich hier nur die wichtigsten Aspekte meines Programms ansprechen. Um diese jedoch genauer zu erklären fehlt dennoch Platz.

Der Gedanke zu „Formula“ entstand bereits in der zehnten Klasse, noch bevor mir die Möglichkeit einer Besonderen Lernleistung bekannt war. Damals versuchte ich gerade, einige mathematische Grundfunktionen, wie zum Beispiel den Arcussinus, algorithmisch darzustellen. Zu dieser Zeit beschäftigte ich mich ebenfalls mit einigen Aspekten der künstlichen Intelligenz. Darunter ist zu verstehen, dass ein Rechner aus den Atomen „Tagsüber scheint die Sonne.“ und „Es ist Tag.“ folgert „Die Sonne scheint.“.

Aus diesen beiden Ideen entstand dann das endgültige Projektziel mit folgenden Aufgabenstellungen.

- Es ist ein Softwareprogramm zu entwickeln welches Aufgaben aus den Bereichen Physik und Chemie lösen kann.
- Dabei sollen Formeln aus einer Datenbasis verwendet werden. Diese Formeln müssen vom Benutzer modifiziert werden können.
- Das Programm soll für jedes Problem selbständig die richtigen Formeln auswählen und ineinander einsetzen.
- Das Programm muss möglichst einfach zu bedienen sein. Daraus folgt, dass Lösungen verständlich formuliert werden und auch ein nachvollziehbarer Rechenweg inklusive Zwischenergebnissen angezeigt wird.
- Jegliche interne Berechnung muss numerisch stabil verlaufen.
- Bei Gleichungen mit mehreren Lösung müssen auch alle Lösungen ausgegeben werden.

Daraus entstand schließlich ein Softwareprogramm für PC-kompatible Computer unter dem Betriebssystem Microsoft Windows (32 Bit). „Formula“ dient dazu, physikalische und chemische Berechnungen zu vereinfachen. So können Aufgabenstellungen direkt in das Programm eingegeben werden. Daraus wird dann automatisch ein Rechenweg ermittelt, und alle notwendigen Gleichungen werden ausgerechnet. Dabei ist es möglich, dass mehrere Formeln ineinander eingesetzt werden.

Der Benutzer erhält eine Lösung der Aufgabe, die allen schulischen Standards („Gegeben“, „Gesucht“ und „Lösung“) entspricht. Selbstverständlich werden auch alle Zwischenergebnisse berechnet und ausgegeben. Die Lösungen können gespeichert oder in die Zwischenablage kopiert und später in ein Dokument eingefügt werden. Auch kann man sie sofort drucken.

Vom Benutzer können neue Formeln eingegeben werden. Ebenso kann man die bereits vorhandenen Formeln bearbeiten oder entfernen.

Das Programm ist auch in der Lage, mehrere Ergebnisse zu berechnen, da beispielsweise bei Dreieckskalkulationen mehr als eine Lösung möglich sein kann. In solchen Berechnungen ist es meist notwendig, auf Winkelfunktionen zu zugreifen. Dabei stehen vier Modi der Winkelangabe (Altgrad, Neugrad, Bogenmaß und Vollkreise) zu Verfügung.

Gute Beispiele für die Zeitersparnis, die die Arbeit mit Formula mit sich bringen kann, finden Sie im [Anhang A](#).

Für die mathematische Seite des Projekts hatte ich mir bereits die Aufgabe gestellt, dass alle Operationen numerisch stabil ablaufen sollen. Dies bedeutet, dass zum Beispiel Fehler wie „Division by Zero“ (englisch für Division durch Null) umgangen werden.

Zusätzlich musste es eine Möglichkeit geben, Formeln dauerhaft abzuspeichern, da es sinnlos wäre, wenn man bei jedem Programmstart alle, möglicherweise notwendigen, Formeln eingeben müsste. Zu all dem kam hinzu, dass ich mir fest vorgenommen hatte, mein Ziel aus eigener Kraft zu erreichen, also keinen der Bestandteile meines Programms aus irgendeiner Literatur zu übernehmen. Einzig beim mathematischen Teil des Projektes war Hilfe notwendig, da mir auf diesem Gebiet noch Kenntnisse fehlen.

Daraus ergaben sich folgende Probleme:

1. Wie stellt man eine mathematische Formel auf dem Computer dar?
2. Wie speichert man solche Formeln dauerhaft?
3. Wie findet man Lösungen für Problemstellungen?
4. Wie rechnet man Formeln mit Hilfe mathematischer Funktionen aus?
5. Wie stellt man die mathematischen Grundfunktionen numerisch stabil dar?
6. Wie schafft man es, alle möglichen Lösungen zu beachten?
7. Wie erstellt man eine sinnvolle, möglichst selbsterklärende, Bedienoberfläche?

Um diese Probleme zu lösen, standen mir zur Verfügung:

1. Die Hilfe von Dr. Gerd Kunert, der mich seit 1999 auf dem Gebiet der Mathematik tatkräftig unterstützt hat.
2. Die Hilfe von Herr König, der mich bei dieser theoretischen Ausarbeitung unterstützt hat.
3. Turbo Pascal 7.0
4. und später Delphi 4.0 Standard

## 2 Wie stellt man eine mathematische Formel auf dem Computer dar?

Damit Formula Aufgaben lösen kann, muss es Formeln ausrechnen können. Zuerst sollen also Formeln vom Papier auf den Rechner übertragen werden.

Um dies zu erreichen, ist es zunächst notwendig, sich über den Aufbau physikalischer Formeln (wie beispielsweise „ $e=mc^2$ “) klar zu werden.

Formeln liefern ein **Ergebnis** zurück und enthalten fast immer **Parameter**, welche verschiedene Werte annehmen können. Das einfachste Beispiel dafür liefert die Mathematik mit einer simplen linearen Funktion.

$$y = x$$

Eine solche Formel darzustellen wäre allein nicht schwierig, aber auch nicht sehr zweckmäßig, da die wenigsten Gleichungen so simpel sind. Sie enthalten dagegen häufigst noch **Rechenoperatoren**, wie man anhand der einfachsten quadratischen Funktion erkennen kann.

$$y = x^2$$

Oft sind auch **Konstanten** enthalten, also feste Werte, die sich nie ändern. Ebenfalls können **Zahlen** enthalten sein.

$$y = -x^2 + 2 * Pi$$

Schließlich können in einer Formel beliebig viele Parameter mit je beliebig vielen Rechenoperationen und ebenso unbegrenzt vielen Konstanten enthalten sein.

$$T = 2 * Pi * Wurzel ( J / (m * g * a) )$$

Hier stellt man einen neuen Bestandteil fest: die **Klammern**, welche weder Parameter, Operatoren, Zahlen oder Konstanten sind. Sie sind lediglich so etwas wie Präfixe, die die Rangfolge der nächsten Operatoren festlegen. Weiterhin dienen Klammern auch zum Begrenzen der Menge der Parameter einer mathematischen Funktion:

$$y = \sin ( x ) + 3$$

Der Sinus wird nur aus x berechnet, während „3“ kein Parameter der Sinusfunktion ist.

Aus rechentechnischer Sicht können nun mehrere Formen mathematischer Funktionen hinsichtlich der Stellung ihrer Operatoren unterschieden werden. Es gibt solche, die vor dem Parameter stehen, wie das Minus in - x, und solche, die zwei Parameter miteinander verknüpfen (m \* g). Die dritte Gruppe von Funktionen steht hinter ihrem Parameter, zum Beispiel  $x^2$ . Im Gegensatz dazu werden in der Mathematik beispielsweise unäre und binäre Operatoren auf Grund der Zahl ihrer Parameter unterschieden.

## 2 1 Das Formelobjekt

---

Eine solche Formel ist nun schon ziemlich kompliziert zu modellieren. Doch nun soll das Modell auf Programmcode übertragen werden. Auch sollen Formeln später im Programm miteinander interagieren können.

An dieser Stelle möchte ich die zwei unterschiedlichen Möglichkeiten, Datenstrukturen in Delphi und Paskal zu kreieren, erklären. Auf der einen Seite sind die Records, welche eine einfache Ansammlung von Variablen darstellen. Auf der anderen Seite stehen die Objekte, ein Verbund von Algorithmen und Daten. Einerseits verfügen sie, ebenso wie Records, über Variablen. Andererseits besitzen sie auch Methoden, mit denen auf diese Variablen zugegriffen werden kann. Deshalb sind sie besonders für die Verwaltung polymorpher (vielgestaltiger) Daten geeignet.

Eine Formel enthält beliebig viele **Parameter**, **Rechenoperatoren**, **Konstanten**, **Zahlen** und **Klammern**. Außerdem sollen die Formeln noch vom Benutzer eingegeben werden, weswegen aus möglichst wenig zur Verfügung stehenden Daten möglichst viel Information gewonnen werden soll. Aus diesen Gründen ist es sinnvoll, Formeln mit Hilfe eines Objekts zu Verwalten.

Um ein solches Objekt zu programmieren, müssen Formeln zunächst auf ihre Bestandteile hin untersucht werden.

## 2 2 Der Titel einer Formel

---

Der erste Bestandteil einer Formel ist der **Titel**, er enthält die Information über das Ergebnis der Formel. Die Funktion des Titels soll anhand der Geschwindigkeit erklärt werden.

$$v = s / t$$

Man kann also dem Titel entnehmen, dass die Formel zur Berechnung der Geschwindigkeit verwendet wird. Um den Titel darzustellen genügt eine simple Zeichenkette (String).

Was passiert jedoch, wenn das Programm  $v$  bestimmen soll, die Datenbasis jedoch auch die Formel

$$v = a * t + v_0$$

enthält? In diesem Fall liegen zwei Formeln mit identischem Titel, jedoch verschiedener Gültigkeit vor. Die erste gilt nur für *gleichförmige* Bewegung, die zweite für *beschleunigte* Bewegung. Das Programm würde also Gefahr laufen, die falsche Formel zu benutzen. Dieses Problem wird gelöst, indem dem Titel einer Formel eine Länge von bis zu 255 Zeichen zugewiesen wird. Er kann beliebige Zeichen (außer Steuerzeichen und den später erklärten Rechen- und Trennzeichen) enthalten. Dies eröffnet dem Benutzer die Möglichkeit, Formeln exakter zu bezeichnen und so auch das Verständnis späterer Rechenergebnisse zu erleichtern. So könnten folgende Formeln besser als

$$\text{Geschwindigkeit der gleichförmigen Bewegung} = s / t$$

und

$$\text{Geschwindigkeit der beschleunigten Bewegung} = a * t + v_0$$

bezeichnet werden. Deshalb erkennt der Benutzer bei einer Berechnung schneller den Sinn seiner Ergebnisse und kann komplizierten Kalkulationen leichter folgen. Dadurch wird es auch einfacher, Probleme zu definieren. Man muss nicht mehr die Formelzeichen beherrschen, um komplizierte Aufgaben zu lösen. Es reicht stattdessen, sie mit einfacher Sprache zu benennen. Wird dieses Verfahren konsequent auf alle Formeln angewandt, erhält der Benutzer stets die gewünschten Gleichungen.

## 2 3 Der Körper einer Formel

---

Der wohl wichtigste Bestandteil einer Formel ist ihr **Körper**. Er sagt aus, welche Rechenzeichen, Parameter, Konstanten und Klammern eine Formel enthält, und wie diese angeordnet sind.

$$\text{Geschwindigkeit der beschleunigten Bewegung} = a * t + v_0$$

Gleichzeitig entsteht auch eines der größten Probleme im Zusammenhang mit Formeln. Der Benutzer soll nur **Formeltitel** und **Formelkörper** eingeben müssen, da alle Parameter bereits im Körper der Formel enthalten sind. Ihn die Parameter eingeben zu lassen würde bedeuten, Daten doppelt

einzelnen. Außerdem müssten diese dann doppelt gespeichert werden und würden unnötig Platz verschwenden. Aus diesen Gründen müssen also alle zur Berechnung notwendigen Informationen aus dem Körper und dem Titel einer Formel gewonnen werden.

Zusätzlich ist es notwendig, den Formelkörper auf seine mathematische und sequenzielle Richtigkeit zu prüfen, da diese durch den Nutzer nicht gewährleistet ist.

Zur Darstellung dient erneut eine Zeichenkette, diesmal mit annähernd unbegrenzter Länge ( $2^{63}$  Zeichen), welche nochmals die genaue, der gebräuchlichen Sprache entnommene, Bezeichnung der Parameter ermöglicht. Auch sind alle Zeichen (außer Steuerzeichen) anwendbar, sofern sie die Logik nicht verletzen. Für das obige Beispiel ergibt sich:

Geschwindigkeit der beschleunigten Bewegung = **Beschleunigung \* Zeit + Anfangsgeschwindigkeit**

Wie man sehen kann, werden dadurch die Formeln länger, weshalb später eine ausgefeilte Eingabemaske benötigt werden wird. Prinzipiell sind nun die wichtigsten Komponenten einer Formel gesammelt. Titel und Körper können so modelliert werden, dass sie vom Benutzer leicht zu verstehen und später auch zu ändern sind. Natürlich ist es ersichtlich, dass der Computer diese Formeln noch nicht berechnen kann. Es wird also noch eine andere Form der Darstellung benötigt. Doch zum jetzigen Zeitpunkt ist es unklug sich mit diesem Problem zu befassen, da noch keine mathematischen Funktionen programmiert sind. Das Formelobjekt kann ja später erweitert werden.

## 2 4 Die **Informationsdaten** einer Formel

---

Um die Übersichtlichkeit und Verständlichkeit der Formeln noch einmal zu verbessern, wird ihnen ein **Kommentar** hinzugefügt. Er soll es ermöglichen, Formeln die sich nicht selbst erklären, zu verstehen.

Geschwindigkeit der beschleunigten Bewegung = Beschleunigung \* Zeit + Anfangsgeschwindigkeit

**Kommentar:** Die Geschwindigkeit eines Körpers ändert sich, von einer bestimmten Anfangsgeschwindigkeit beginnend, pro Zeiteinheit um einen festen Wert, welchen die Beschleunigung angibt. Diese Formel dient der Berechnung seiner Geschwindigkeit zu einem, in „Zeit“ festgelegten, Zeitpunkt.

**Kommentare** können, ebenso wie der Formelkörper, fast beliebig lang sein. Es gibt für die enthaltenen Zeichen keinerlei Vorschriften.

Zuletzt wird für jede Formel noch das **Datum** gespeichert, an dem sie erstellt wurde. Dies gibt beispielsweise für Schüler Auskunft über ihren Lernprozess, wenn regelmäßig neu angelegte Formeln eingegeben werden.

## 3 Wie speichert man solche Formeln dauerhaft?

---

Die im **zweiten Kapitel** entwickelten Formelobjekte sollen nun in einer Art Datenbasis (der Einfachheit halber „Formelbasis“) dauerhaft gespeichert werden. Dabei muss berücksichtigt werden, dass dies schnell geschehen soll. Überdies sollen die Formeln auch schnell wieder eingelesen werden können, was noch wichtiger ist.

Um aber überhaupt die Interaktionen mit einer Datei fehlerfrei und effizient zu gewährleisten, bringt die Entwicklung eines Dateiobjekts Vorteile. Dieses Objekt verwendet direkt die von Windows zur Verfügung gestellten Dateifunktionen.

### 3 1 Das TFile-Objekt

---

Das TFile-Objekt bildet die Grundlage für die beiden formelorientierten Dateien der Formelbasis. Es kapselt die Windows-Dateihandles und stellt die notwendigen Funktionen im Umgang mit Dateien zur Verfügung. Auch besitzt es verschiedene Über- und Unterlaufsperrungen, welche Fehler im Zugriff verhindern.

Die „IsOpen-Sperre“ verhindert zum Beispiel, dass versucht wird, in eine geschlossene Datei zu Schreiben, daraus zu Lesen sowie alle anderen ähnlichen Tätigkeiten.

Die „Position and Size-Sperre“ wird beim Öffnen einer Datei initialisiert, wenn die „IsOpen-Sperre“ wegfällt. Dafür besitzt das Objekt die Eigenschaften „Position“ und „Size“, welche die aktuelle Position in der Datei und deren Größe speichern. Die Funktion der Sperre wird an einem Beispiel deutlich:

Angenommen, eine geöffnete Datei ist 200 Bytes groß und die Lese- und Schreibmarke befindet sich nach dem 100. Byte. Nun wird versucht, die Position der Lese- und Schreibmarke 200 Bytes nach hinten zu versetzen. Dann erkennt die Sperre, dass dies hinter dem Dateiende liegen würde und versetzt sie nur um 100 Bytes, also auf das Dateiende. Würde diese Sperre nicht existieren, hätte Windows die Datei entsprechend vergrößert, also auf 300 Bytes, und diese würden nun undefinierte Daten enthalten. Ebenso wird die Sperre bei Lesezugriffen genutzt. Auf Schreibzugriffe hat sie keinen Einfluss.

Die „Share-Sperre“ kommt ebenfalls schon beim Öffnen der Datei zum Zuge, sie verhindert, dass die selbe Datei mehr als einmal geöffnet wird, um Komplikationen mit anderen Programmen zu vermeiden.

Zusätzlich verfügt das Objekt auch über Funktionen, mit denen Zahlen und Zeichenketten gelesen und geschrieben werden können. Natürlich kommen auch hier wieder die Sperren zum Einsatz um den korrekten Umgang mit Daten zu sichern. Beim Lesen von Zeichenketten gibt es noch zwei weitere Sperren. Eine verhindert, dass Zeichenketten eingelesen werden, die mehr Arbeitsspeicher benötigen als vorhanden ist. Die andere, die Stringlängensperre, kommt besonders im Umgang mit Formeln zum Tragen. Sie lässt nur Zeichenketten zu, die eine angegebene Länge nicht überschreiten.

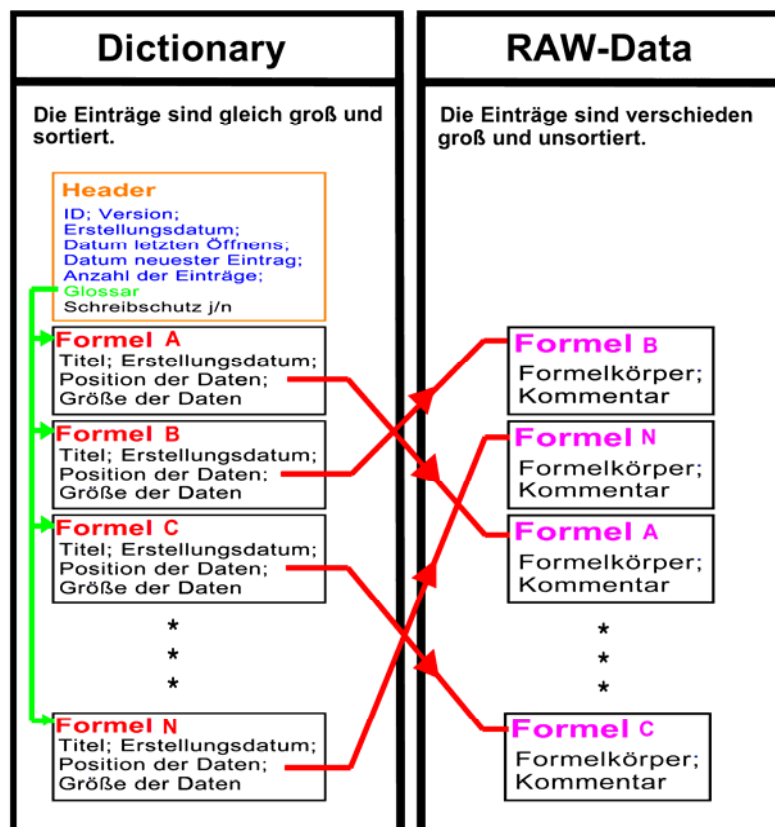
### 3 2 Das Dictionary

Wie bereits im [Kapitel zwei](#) ersichtlich ist, können Formeln polymorphe Daten enthalten, was bedeutet, dass sie auch verschieden viel Speicher belegen. Würde man die Formeln hintereinander in eine Datei schreiben, so entstünden zwei große Probleme. Befinden sich die Formeln innerhalb der Datei in alphabetischer Ordnung, so muss diese bei jedem neuen Eintrag fast komplett neu erstellt werden. Sind die Formeln jedoch ungeordnet und es soll eine bestimmte Formel geladen werden, so muss die ganze Datei durchsucht werden. Beide Male ist mit enormer Geschwindigkeitseinbuße zu rechnen. Datenbasen die aus verschiedenen großen Einheiten bestehen, haben eine größere Zugriffszeit. Es erscheint also logisch, eine Formel auf zwei Dateien aufzuteilen: eine, in der alle Daten stehen, die immer gleich groß sind und eine, in der alle variablen Daten gespeichert sind.

Aus diesem Grund entstand das Dictionary (engl. für Wörterbuch), welches den Grundstein für den schnellen Zugriff auf die Daten bildet. Beim Öffnen einer Formelbasis wird es deshalb in den Speicher geladen. Durch die 32-Bit-Speicheradressierung in Microsoft Windows ist die Anzahl der Einträge einer Formelbasis auf 7.669.582 beschränkt. Da eine solche Formelbasis aber zu groß für den effizienten Gebrauch wäre, und es meiner Meinung nach auch gar nicht so viele Formeln auf einem Gebiet gibt, spielt diese Beschränkung praktisch keine Rolle.

Dictionary-Dateien werden durch die Dateiendung (Suffix) „.tsd“ gekennzeichnet und befinden sich stets im selben Verzeichnis wie die übrigen Dateien der Formelbasis.

Am Anfang des Dictionary steht ein **Header**, also ein spezieller Record, der **Informationen** über die gesamte Formelbasis enthält. Dieser Header enthält die Kennung einer Formelbasis (ID), deren Version (zur Zeit noch 1), ihr Erstellungsdatum sowie das Datum des letzten Öffnens und das des letzten Schreibzugriffes (und somit das des neusten Eintrags).



Darauf folgt der **Glossar**. Der Glossar ist ein Array, also eine Liste, der für jedes mögliche Anfangszeichen einer Formel eine Zahl (Integer) enthält. Er beschleunigt den auf Zugriff die Daten, viele Lesezugriffe werden gespart. Soll eine Formel geladen werden, zum Beispiel die Formel für die „Zugkraft“, so wird das Anfangszeichen („Z“) in den Glossar eingesetzt und man erhält die Position im Dictionary, ab der alle Einträge, die mit „Z“ beginnen, aufgelistet sind. Um den Eintrag für die Zugkraft zu finden, wird das Dictionary nun linear ab dieser Position durchsucht, bis dieser entweder entdeckt, ein anderer Eintrag gefunden wurde, der alphabetisch dahinter steht, wie beispielsweise „Zugwinkel“, oder das Ende des Dictionarys erreicht wurde.

Auf den Header folgen nur noch **Datensätze der gleichen Größe**. Ein Eintrag setzt sich aus dem Titel der Formel (deshalb sind nur 255 Zeichen zugelassen), ihrem Erstellungsdatum und dem Verweis auf die Datei mit **variablen Daten** (Raw-Data-Datei; Suffix “.dbq”) zusammen. Dieser Verweis besteht aus der Position der Daten und deren Größe. Bei dem Beispiel für die Zugkraft wird also der passende Formelkörper und das dazugehörige Kommentar sofort gefunden und kann aus der Raw-Data-Datei gelesen werden.

Die Größe der variablen Daten ist beim Laden und Speichern irrelevant und ergibt sich automatisch aus der Länge der Zeichenketten. Sie wird aber dennoch gespeichert, um das Löschen von nicht mehr benötigten Formeln zu beschleunigen und beim Laden eine weitere Sperre, die Stringlängensperre (siehe [Kapitel 3.1](#)), zu ermöglichen. Es wird stets geprüft, ob die Länge der Strings zusammen auch wirklich die angegebene Größe ergeben. Wenn nicht wird der Rest abgeschnitten.

---

### 3 3 Das Speichern einer Formel

---

Mit diesen vordefinierten Eigenschaften einer Formelbasis ist das Speichern einer Formel sehr simpel. Zuerst wird ein Eintrag für das Dictionary im Speicher erstellt und mit den erforderlichen Werten belegt. Anschließend wird er in das Dictionary eingefügt. Dann werden die variablen Daten der Formel an das Ende der Raw-Data-Datei angehängt. Damit ist die Formel für spätere Verwendung gesichert. Zu bemerken wäre vielleicht noch, dass Änderungen des Dictionary im Speicher erfolgen und erst beim Schließen der Formelbasis auf die Festplatte übertragen werden.

---

## 4 Wie findet man Lösungen für Problemstellungen?

---

Formula dient dazu, Aufgaben zu lösen. Dabei soll das Programm im Prinzip ebenso wie ein Mensch verfahren. Zuerst muss also aus allen bekannten Formeln die Menge der benötigten Formeln ausgewählt werden. In diesem Kapitel werde ich also eine Methode herleiten, die aus einer Formelbasis die für eine Berechnung benötigten Formeln selektiert.

Will der Benutzer eine Aufgabe lösen, so wird er später an einer Stelle der Bedienoberfläche den Namen der gesuchten Größe angeben. Dieser Name entspricht dem Titel einer Formel.

Um gegebene Probleme lösen zu können, ist es notwendig, die zu diesem Titel passenden Formel zu finden, womit sich [Kapitel 4.1](#) beschäftigen wird. In der Lösung können auch verkettete Formeln vorkommen, so dass dafür ein effizienter Algorithmus gefunden werden muss. Mit der Suche nach diesem Algorithmus werde ich mich in den [Kapiteln 4.2](#) bis [4.4](#) beschäftigen.

---

### 4 1 Formeln finden

---

Um die korrekte Arbeitsweise der Formelbasis und des Formelobjekts zu prüfen, wurde zu diesem Zeitpunkt eine Testoberfläche erstellt. Ein Teil dieser Testoberfläche war eine Tabelle, in der alle Formeln einer Formelbasis ihrem Titel nach alphabetisch aufgelistet waren. Eine solche Tabelle nennt man in der Informatik Hashtabelle. Es war bereits klar, dass auch die entgeltige Bedienoberfläche eine ähnliche Tabelle enthalten würde. In der rechten Spalte standen alle Titel der Formeln, in der linken die dazugehörigen Körper.

Um eine Formel zu lösen, benötigt der Parser weder den Kommentar noch das Erstellungsdatum der Formel. Alle Informationsdaten fallen also weg. So bleiben lediglich Titel und Körper einer Formel übrig.

Der Arbeitsspeicher eines normalen Computers ist etwa 1'000'000mal so schnell wie eine Festplatte. Es würde also Performancegewinn (Beschleunigung des Programms) mit sich bringen, wenn Formeln direkt aus dem RAM (Arbeitsspeicher) erstellt würden, anstatt sie von der Festplatte zu laden.

Alles was nötig ist, um eine lösbare Formel zu erstellen, befindet sich in der oben genannten Tabelle in der Testoberfläche und somit im Arbeitsspeicher. Versieht man diese Tabelle mit Methoden, um zu

einem gegebenen Titel die passende Formel auszusuchen, so kann man gegenüber dem Suchen in der Formelbasis viel Zeit sparen.

Es fehlt also nur eine Möglichkeit, aus einer Menge dem Alphabet nach sortierter Daten ein bestimmtes Datum auszusuchen. Die erste Möglichkeit ist, von Beginn der Tabelle an, alle Formeltitel mit dem gesuchten Titel zu vergleichen. Bei einer Tabelle mit  $n$  Einträgen entspräche dies einem Suchaufwand von maximal  $n$  Vergleichen. Weiterhin eignete sich der Hash-Suchalgorithmus, welcher mit einem maximalen Suchaufwand von  $\log_2 n + 1$  Vergleichen weit effektiver ist. Dieser lässt sich an einem einfachen Kode-Stück gut erklären.

```

Const Maximal      = 100;

Type  TabelleType  = Array[1..Maximal] of String;

Function Find(Such  :      String;
              Tabelle: TabelleType): Integer;

Var H: Integer;

Begin

Result := Maximal div 2;

H      := Result div 2;

While (Tabelle[Result] <> Such) And
      (Result > 0)           And
      (Result <= Maximal)   do

  Begin

    If ImAlphabetDavor (Such, Tabelle[Result])
      Then Result := Result - H
      Else Result := Result + H;

    If H > 2 Then H := H div 2
      Else H :=      1;

  End;

If Tabelle[Result] <> Such Then Result := 0;
End;

```

Hier wird eine **Liste von Strings** definiert, welche so viele Einträge besitzt, wie der Wert von **Maximal** angibt (also 100). Sie ist stets eine Hashtabelle und somit alphabetisch sortiert.

Die **Funktion Find** soll aus einer solchen Liste (**Tabelle**) die Position eines bestimmten Strings (**Such**) zurückliefern. Die **Variable H** ist eine Ganzzahl und soll später immer die Schrittweite, mit der der interne Zähler vergrößert oder verringert wird, enthalten.

**Result** ist eine Variable über die jede Delphi-Funktion verfügt und die deren Ergebnis beinhaltet. Am Beginn der Funktion werden also **Result** und **H** mit ihren Startwerten initialisiert.

Da die Einträge in der **Tabelle** wie in einem **Telefonbuch** sortiert sind, lässt sich der folgende Ablauf so erklären: Das Telefonbuch wird **in der Mitte aufgeschlagen und gefundene Eintrag mit dem gesuchten verglichen**. Es gibt drei Möglichkeiten: entweder er entspricht diesem, so hat man ihn gefunden oder **er steht im Alphabet davor bzw. danach**. Steht er davor, so beginnt man von vorn, jedoch **nur mit der ersten Hälfte des Telefonbuchs**, anderenfalls mit der letzten Hälfte. Dies tut man so lange, bis man den gesuchten Eintrag entweder gefunden hat (d.h. das der **Vergleich am Ausgangspunkt der Schleife „wahr“** ergibt) bzw. am **Anfang oder am Ende des Telefonbuchs angekommen ist**.

Steht der Eintrag nicht in der **Liste**, so wird **Null als Ergebnis** geliefert.

Mit dieser Methode kann das Programm also recht schnell eine Formel mit dem passenden Titel zu einem Problem finden. Da es zu einem Titel aber mehrere Formeln geben kann, wird von der gefundenen Position ab, bis zum ersten Auftreten des Titels, rückwärts gesucht. Denn mit der Hashsuche wird nicht zwangsläufig dessen erstes Vorkommen zurückgegeben.

Jetzt sollte der Suchalgorithmus noch mit den speziellen Bedürfnissen des Programms verbunden werden. Es wird unterschieden zwischen verschiedenen Sorten von Formeln, deren Titel mit dem gesuchten Wert übereinstimmt und die für die Lösung eines Problems in Frage kommen.

Angenommen, es sind die Werte für die Parameter **X**, **Y** und **Z** bekannt. Gesucht ist eine Formel, mit deren Hilfe sich der Wert von „Q“ berechnen lässt. Diese Formel besäße also den Titel „Q“. In einer Formelbasis können drei Mengen von Formeln mit diesem Titel existieren:

1. Die Formeln des ersten Typs besitzen exakt die bekannten Parameter. Sie verfügen genau über die Parameter **X**, **Y** und **Z**.

2. Eine zweite Menge von Formeln besitzt andere Parameter als die bekannten, welche vielleicht aus diesem berechnet werden könnten. Eine solche Formel weist beispielsweise die Parameter **A**, **B** und **Y** auf.
3. Die Klasse mit dem geringsten Wert für Lösungen besitzt nur einige der bekannten Parameter, jedoch nicht alle. Dies würde bei einer Formel mit den Parametern **X** und **Z** zutreffen.

Um die Formeln den oben genannten Mengen zuordnen zu können, wird das Formelobjekt um einige Methoden erweitert. Diese ermöglichen es ihm, seine Parameter mit einer anderen Menge von Parametern, zu vergleichen.

## 4 2 Der Lösungsbaum

Sind Formula die Namen einer gesuchten und einiger bekannter Größen gegeben, so findet es die passende Formel. Aufgaben erfordern es jedoch oft, Formeln ineinander einzusetzen. Dafür muss nun eine Methode hergeleitet werden.

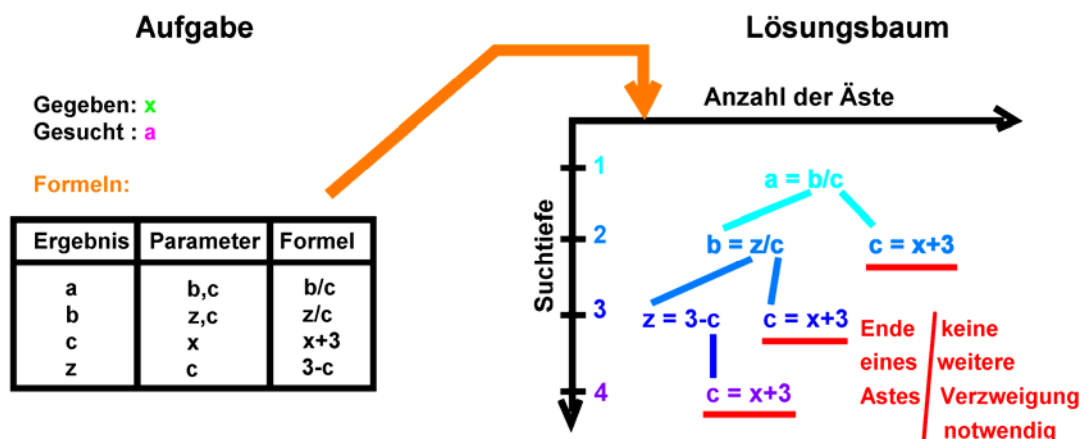
Um einen Algorithmus zu entwickeln, der Formeln sucht und ineinander einsetzt, ist es sinnvoll, sich zuerst ein Modell dafür zu erstellen. Ein grafisches Modell von dem angestrebten Ablauf ist am einfachsten mit Hilfe eines simplen Beispiels anzufertigen.

Angenommen, in der **Formelbasis** befinden sich die Gleichungen „ $a = b/c$ “, „ $b = z/c$ “, „ $c = x+3$ “ und „ $z = 3-c$ “. Gesucht ist ein Wert für „ $a$ “ und bekannt sei die Größe „ $x$ “.

Das Programm verfügt nun über alle notwendigen Daten, um die Aufgabe zu lösen.

Es soll nun zuerst eine Formel finden, die „ $a$ “ direkt aus „ $x$ “ berechnen kann. Da eine solche Formel in der Formelbasis nicht existiert, wird als nächstes eine Formel für „ $a$ “ gesucht, die über andere Parameter verfügt. Diese anderen Parameter müssen später berechnet werden. Deshalb müssen für sie ebenfalls passende Formeln gefunden werden. Mit diesen wird dann wie mit „ $a$ “ verfahren. Aufgehört wird erst, wenn sich alle Parameter der Formel für „ $a$ “ berechnen lassen oder aber keine weiteren passenden Formeln für die Parameter gefunden werden. In diesem Fall würde nach einer weiteren Formel für „ $a$ “ gesucht, und da diese nicht existiert, beendet.

Eine Aufgabe setzt sich aus der Menge der gegebenen Größen, der Menge der bekannten Formeln und der gesuchten Größe zusammen. Dies kann als Tabelle dargestellt werden, während die Auswahl der benötigten Formeln eher einem Baumdiagramm entspricht.



Hier kann man erkennen, dass die Formel „ $c = x+3$ “ dreimal benötigt wird. Es wäre umständlich, „ $c$ “ dreimal zu „finden“. Deshalb sollen alle Titel der Formeln eines Astes, dessen Ende erreicht wurde, als gegebene Größen betrachtet werden. Dadurch braucht auch für den zweiten Parameter der Formel für „ $a$ “ (nämlich „ $c$ “) keine Formel mehr gesucht zu werden, da diese dann bereits bekannt ist.

Bei der Formelfindung muss auch beachtet werden, ob Formeln sich von einander ableiten. Ein Problem, welches eine große Rolle spielt. So könnten zum Beispiel die Formeln „ $a = b+c$ “ und „ $b = a-c$ “ in der Formelbasis enthalten sein.

Angenommen, bei einer Fragestellung ist nun der Parameter „c“ gegeben, nicht aber der Parameter „b“. Wäre „a“ gesucht, so würde das Programm in eine Endlosschleife geraten. Zuerst würde eine Formel für „a“ gesucht („ $a = b+c$ “), wofür ein Wert für „b“ benötigt wird („ $b = a-c$ “), weswegen ein Wert für „a“ gesucht werden müsste („ $a = b+c$ “) ...

Dies wird vermieden, indem beim Suchen neuer Formeln immer zuerst die Menge aus allen Parameternamen und dem Titel der Formel gebildet wird. Ebenso wird mit der im Lösungsbaum darunter stehenden Formel verfahren. Anschließend werden beide Mengen verglichen. Sind sie identisch, so sind die Formeln lediglich umgestellte Versionen von ein und dem selben Term und können nicht verwendet werden.

## 4 3 Rekursion

---

Zum Zeitpunkt der Aufgabenstellung ist noch unbekannt, wie die Formeln ineinander eingesetzt werden müssen. Auch kann deren Anzahl von Aufgabe zu Aufgabe verschieden sein. Bekannt ist jedoch, dass mit jeder einzusetzenden Formel im Prinzip gleich verfahren werden muss. Für jeden ihrer Parameter, der nicht bekannt ist, muss eine Formel gefunden und eingesetzt werden. Es ist also eine Technik zu suchen, die es ermöglicht, für eine unbekannte Anzahl von Durchläufen, Formeln einzusetzen und nach unbekanntem Parametern zu durchsuchen.

In der Fachliteratur wird nur eine solche Programmier Technik erwähnt, und auch mir ist nur diese eine bekannt. Diese Technik wird als Rekursion bezeichnet. Sie wird auch in Simulationsprogrammen und vielen Spielen verwendet. So werden mit ihrer Hilfe zum Beispiel beim Schachspiel Züge vorausgerechnet.

Die Arbeitsweise rekursiver Prozeduren ist relativ simpel. Sie rufen sich selbst auf, bis ein Abbruchkriterium erfüllt ist. Ein simples Beispiel bietet die mathematische Funktion „Fakultät“. Diese erhält als Parameter eine natürliche Zahl. Das Produkt aller natürlicher Zahlen von eins an bis zu dieser gegebenen Zahl wird als deren Fakultät bezeichnet. Als rekursive Funktion umgesetzt könnte die sie etwa so dargestellt werden:

```
Function Fakultaet(I: Integer): Integer;  
  
Begin  
If I = 1 Then Result := 1  
Else Result := I * Fakultaet(I-1);  
End;
```

Die Funktion **Fakultaet** soll die Fakultät der natürlichen Zahl **I** zum Ergebnis haben. Wenn **I eins entspricht**, so ist das Ergebnis auch eins, anderenfalls entspricht es der Fakultät von **I – 1** multipliziert mit **I**. Die Funktion ruft sich selbst auf, solange das Abbruchkriterium **I = 1** nicht erfüllt ist.

Der wahre Nutzen von Rekursion liegt in der Kürze des Programmiercodes und gleichzeitig darin, dass sie auch oft schneller als vergleichbare Schleifen ist. Sie wird vor allem bei Aktionen angewendet, die, graphisch dargestellt, etwa ein Baumdiagramm ergeben. Im [vorigen Kapitel](#) wurde ersichtlich, dass der Formelfindungsprozess als solch ein Baum modelliert werden kann.

## 4 4 Die Formelsequenz

---

Im [vorherigen Kapitel](#) wurde erklärt, wie geeignete Formeln für eine Aufgabe gefunden werden können. Das so erstellte Modell muss nun in die Praxis umgesetzt werden. Auch soll herausgefunden werden, in welcher Reihenfolge die so ausgesuchten Formeln berechnet werden müssen.

Diese Reihenfolge, oder auch Formelsequenz, kann bei jeder Aufgabe verschieden sein. Außerdem können zur Lösung jedes Problems unterschiedlich viele Formeln benötigt werden.

In Delphi werden bereits Listen mit beliebiger Länge zur Verfügung gestellt, die dynamischen Arrays. Aus diesem Grund wird die Formelsequenz als dynamischer Array im Speicher erstellt. Sie entspricht der umgekehrten Reihenfolge, in der die Formeln des Formelbaums gefunden wurden.

Wird also der Lösungsbaum aufgestellt so werden, wie oben erwähnt, alle gefundenen Formeln in eine Liste bekannter Größen eingetragen. Diese Liste rückwärts gelesen entspräche dann bereits der Formelsequenz.

Die Funktion, welche den Lösungsbaum erstellt, ist das wichtigste Modul von Formula. Aus einer Menge gegebener Formeln in der Formelbasis, einer Menge gegebener Werte für Parameter und einem gesuchten Wert findet sie die passenden Formeln in der richtigen Reihenfolge. Wegen ihrer Wichtigkeit soll sie hier als Pseudo-Code beschrieben werden.

Die Funktion hat dem Namen `FindTerm`. Sie erhält bei jedem Aufruf die folgenden Parameter:

Parameter	Typ	Aufgabe
<b>B</b>	Boolean (kann entweder Wahr oder Falsch annehmen)	Sind Formeln der dritten Klasse (siehe <a href="#">Kapitel 4.1</a> ) als Ergebnis zulässig?
<b>T</b>	String (Zeichenkette)	der Name der gesuchten Größe
<b>F</b>	Formel	die Formel welche im Lösungsbaum einen Ast höher steht

Zusätzlich verfügt sie über zwei private Variablen, welche bei jedem Aufruf intern neu erstellt werden. Die natürliche Ganzzahl **N** wird als Zähler für gefundene Formeln verwendet. Die Variable **C** (vom Typ Boolean) hat eine ähnliche Bedeutung wie der Parameter **B**. Sie dient lediglich dazu, unnötiges Mehrfachprogrammieren von Code zu sparen.

Alle bekannten Größen und somit auch die Parameter stehen in einer globalen Liste.

Beim Aufruf der Funktion, welche die passende Formel für das Problem **T** als Ergebnis liefern soll, entspricht **B** wahr und **F** keiner Formel.

1. Finde eine Formel, deren Titel **T** ist und deren Parameter exakt mit allen gegebenen Größen übereinstimmen. Diese Formel entspräche dem ersten Typ aus dem [Kapitel 4.1](#).
2. Existiert diese Formel, liefere sie als Ergebnis, trage sie in die Liste der bekannten Größen ein und breche ab.
3. Die interne Variable **N** erhält den Wert eins.
4. Die interne Variable **C** erhält den Wert Falsch.
5. Finde die **N**-te Formel mit anderen als den gegebenen Parametern die den Titel **T** besitzt. Diese Formel wäre vom Typ zwei.
6. Existiert diese Formel nicht gehe zu Schritt elf.
7. Ist **F** eine definierte Formel (beim ersten Aufruf nicht), so prüfe, ob die gefundene Formel eine umgestellte Version von **F** ist. Wenn ja, erhöhe **N** um eins und gehe zum fünften Schritt.
8. Rufe `FindTerm` (also den Algorithmus selbst) rekursiv für jeden unbekanntem Parameter der gefundenen Formel auf. Setze die Parameter der Funktion bei jedem Aufruf wie folgt:
  - T** = Name des gesuchten Parameters
  - B** = **C**
  - F** = die eben gefundene Formel
9. Konnte für alle Parameter eine Lösung gefunden werden? Wenn nicht, erhöhe **N** um eins und gehe zu Schritt fünf.
10. Trage den Titel der gefundenen Formel als bekannte Größe ein, liefere sie als Ergebnis und breche ab.
11. Ist **C** wahr, gehe zum Schritt 13.
12. Ist **B** wahr, so setze **C** auf wahr und gehe zum Schritt Nummer drei. Anderenfalls breche ab.
13. Finde die Formel, deren Parameter so viele der bekannten Größen wie möglich enthalten. Diese Formel entspricht dem dritten Typ aus dem [Kapitel 4.1](#).
14. Existiert diese Formel so trage ihren Titel als bekannte Größe ein und liefere sie als Ergebnis. Anderenfalls liefere keine Formel als Ergebnis.

Dieser Pseudo-Code kann nur vereinfacht die Komplexität des reinen Quelltextes der Methode wiedergeben. Dennoch enthält er alle wesentlichen Prinzipien, um für eine Aufgabe die bestmögliche Lösung aus einer Menge der gegebenen Parameter und Formeln zu finden. Es wird beachtet, dass zu einem Problem mehrere Lösungsmöglichkeiten existieren können. Davon wird immer diejenige ausgewählt, welche möglichst alle gegebenen Größen einbindet. Ebenso prüft der Algorithmus, ob Formeln bereits gefunden wurden oder sich von einander ableiten.

Dieser Code ist, gegenüber allen meinen anderen Versuchen auf diesem Gebiet, bei weitem der schnellste und kompakteste.

## **5 Wie rechnet man Formeln mit Hilfe mathematischer Funktionen aus?**

Zu diesem Zeitpunkt der Entwicklung Formulas findet das Programm zu einer Aufgabe die richtige Sequenz von Formeln. Damit ist die Aufgabe aber noch nicht gelöst, denn die Formeln müssen noch ausgerechnet werden. Hierfür ist es notwendig, Werte für ihre Parameter einzusetzen. Dazu existieren verschiedene Methoden, die hier gegenübergestellt werden sollen.

Die frühen Formula-Version nutzen zur Berechnung Compiler. Compiler sind Software, die aus

Quelltext ausführbare Programme erstellt.

In diesen Formula-Versionen wurde dazu der Formelcode in Pascal-Kode umgewandelt, um mit Hilfe des Compilers eine ausführbare „EXE“-Datei (engl. executeable für ausführbar) zu erstellen. Diese wurde von Formula gestartet und lieferte dann das Ergebnis zurück. In der Alphaversion, welche noch komplett unter MS-DOS lief, wurde der Turbo Pascal 7.0-Compiler verwendet. Das Ergebnis wurde vom Unterprogramm an einer bestimmten Stelle im Speicher abgelegt.

Die erste Formula Version für Windows (MS Windows 3.11) erstellte die „EXE“-Datei bereits mit dem Delphi 3.0-Compiler und nutzte DDE (engl. dynamic data exchange für dynamischer Datenaustausch) zur Datenübermittlung.

Später wurde der Delphi 3.0-Compiler durch den Delphi 4.0-Compiler ersetzt und anstatt einer „EXE“-Datei eine „DLL“-Datei (engl. dynamic link library für Bibliothek) erzeugt. Diese wurde vom Programm geladen und nach der Berechnung freigegeben. Weil das Programm direkten Zugriff auf die Ergebnisse hatte, konnte nun auf Datenaustausch verzichtet werden,

Waren all diese Methoden auch programmtechnisch leicht umzusetzen und relativ effektiv, so hatten sie doch auch große Nachteile. Es wurde stets ein Programm verwendet, welches nicht mein geistiges Eigentum war. Auch mussten immer mindestens zwei Dateien erstellt und gelöscht werden. Dazu ist das Compilieren und Ausführen bzw. Laden der Unterprogramme relativ langsam. Weiterhin war es nicht möglich, Berechnungen abzubrechen. Bis zu der Nutzung von DLLs wurde sogar die Systemsicherheit durch Formula gefährdet.

Es sollte also eine Methode gefunden werden, die die vom Benutzer eingegebene Abfolge von mathematischen Operationen interpretiert. Die Formeln, also Zeichenketten, sollen in Aufrufe der mathematischen Funktionen, welche Programmcode sind, umgewandelt werden.

Die vom Benutzer eingegebene Form	Die mathematischen Funktionen im PC
$T = 2 * \text{Pi} * \text{Wurzel} ( J / (m * g * a) )$	Function <b>Wurzel</b> (Const A1 : R): R; . . Function <b>Mul</b> (Const A1, A2: R): R; Function <b>Divi</b> (Const A1, A2: R): R; .

Der Unterschied zwischen den beiden Formen wird hier noch nicht deutlich. Man kann ihn am besten an einem Vergleich erklären: Die linke Seite stellt den Bauplan für ein Haus dar, die rechte die Einzelteile des selben. Der Bauarbeiter ist der mathematische Prozessor und das Papier, auf dem der Plan steht, der Arbeitsspeicher. Dies ist die momentane Lage. Der zweidimensionale Bauplan muss nun irgendwie mit Hilfe des Bauarbeiters in ein dreidimensionales Haus umgesetzt werden. Das Problem ist, der Bauarbeiter ist ein Lehrling. Er weis nicht, was er tun soll, geschweige denn, was der Plan bedeutet. Ihm ist auch unbekannt, wie er die verschiedenen Werkzeuge, nämlich die mathematischen Funktionen, benutzen soll.

Der erste Schritt, ihn „anzulernen“ ist eine Schnittstelle zwischen der Anweisung „Nimm die Schaufel und grabe!“ und dem praktischen „die-Schaufel-nehmen-und-graben“ zu finden.

Eine Funktion ist trivial formuliert ein Stück Code, das an irgendeiner Stelle im Arbeitsspeicher beginnt und irgendwo anders endet. Man bezieht sich auf sie über Zeiger, sogenannte „Pointer“, die auf ihren Anfang zeigen. Eine simple Schnittstelle wäre also eine Tabelle, die auf der einen Seite ein Textstück (String), zum Beispiel „**Mul**“ oder „**Divi**“, enthält, und auf der anderen Seite zu jedem Textstück einen Zeiger auf die dazugehörige Funktion. Dazu wird noch gespeichert, wie viele Parameter die Funktion benötigt. Diese Tabelle wird als *Globalfunctiontable* bezeichnet.

Um eine bestimmte mathematische Funktion auszuführen wird nun ihr Bezeichner (zum Beispiel „Wurzel“) in der *Globalfunctiontable* nachgeschlagen. Somit wird ein Zeiger auf die Funktion ermittelt. Dann werden die Parameter der Funktion auf dem Stack (bestimmter Teil des Arbeitsspeichers) abgelegt. Durch die Assembler-Anweisung „CALL“ wird die Funktion aufgerufen und das Ergebnis ermittelt.

An dieser Stelle ist der Rechner in der Lage, den Befehl „Multipliziere!“ zu befolgen, wenn ihm die beiden notwendigen Parameter gegeben sind.

Doch diese sind ihm noch nicht gegeben. Sie stehen als Zahlen oder Parameterbezeichner (z.B. „x“) im Formelkörper.

Als nächster Schritt müssen diese schriftlichen Kommazahlen in Kommazahlens des ANSI/IEEE-Standards umgewandelt werden, damit der Computer mit ihnen rechnen kann. Dies wird mit der Delphi-internen Funktion StrToFloat bewerkstelligt, die Zahlen in den Datentyp mit der höchsten Genauigkeit, „Extended“, umwandeln kann.

## 5 1 Das Modell des Parsing

Um nun die richtige Abfolge der mathematischen Operationen aus dem Text eines Formelkörpers zu ermitteln, soll ein weiteres Verfahren angewandt werden. Es ist weitaus effektiver als die zu Beginn des [fünften Kapitels](#) genannten Methoden.

In diesem Abschnitt möchte ich das Prinzip dieses Verfahrens, das „Parsing“, erklären.

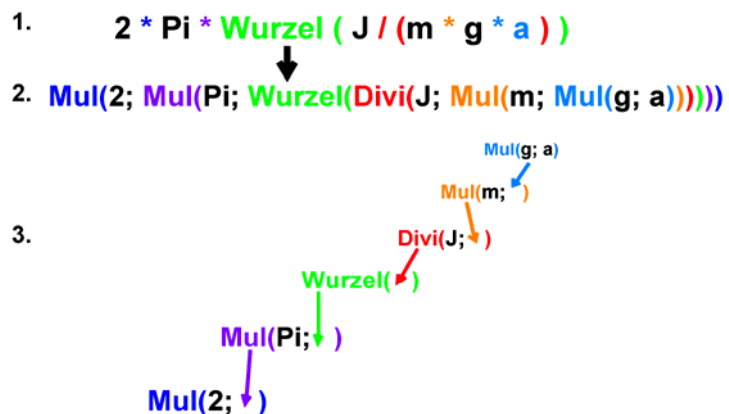
Ein Interpreter ist ein Programm, welches eine Zeichenkette analysiert. Diese besteht meist aus einer Folge von Befehlen. Ist das Interpretieren solcher Strings nur ein Bestandteil des Programms, so wird dieser Bestandteil als „Parser“ bezeichnet. Solche Parser sind zum Beispiel in handelsüblichen Tabellenkalkulationen beziehungsweise Spreadsheets enthalten.

Das Prinzip des Parsing besteht hauptsächlich darin, die Zeichenketten in einzelne Anweisungen zu zerlegen und diese ihrer Reihenfolge nach auszuführen.

Auf das Formelprogramm übertragen bedeutet dies, dass folgender Ablauf entsteht:

1. Die vom Benutzer eingegebene Formel ist eine Zeichenkette. Sie befindet sich in einem Format, in dem die Abfolge (Sequenz) der einzelnen Rechenoperationen nicht ersichtlich ist.
2. Sie wird zunächst in eine andere Form, den „Parserstring“, umgewandelt. Nun ist die Reihenfolge der Operationen und deren Parameter eindeutig. Die neue Zeichenkette ist leichter zu analysieren und die Operationen befinden sich in mathematisch korrekter Reihenfolge.

3. Nun wird der Parserstring ausgerechnet. Dabei wird mit der Operation begonnen, die am tiefsten in Klammern eingebettet ist. Deren Ergebnis wird wie in der Skizze in die nächsthöhere Operation eingesetzt und deren Ergebnis wiederum in die nächsthöhere, bis es keine höhere Operation mehr gibt und das Ergebnis des Terms berechnet wurde.



## 5 2 Der Parserstring

Das komplizierteste am Parsen ist das Umwandeln der vom Benutzer eingegebenen Form der Formeln (Form 1), in eine Form, die der Parser auswerten kann. Darum möchte ich jetzt die Entstehung des Parserstrings (Form 2) erklären.

Dies ist auch die, im [Kapitel 2.3](#) erwähnte, zweite Methode der Darstellung des Formelkörpers.

Zuerst werden sämtliche Rechenzeichen ( $\pm + - * / || ! ^ \% \% \circ ^ 1 2 3$ ) in Funktionsbezeichner (Plusminus, Add, Sub, Mul...) umgewandelt. Es ist aber zu beachten, dass es, wie in [Kapitel zwei](#) erklärt, aus rechentechnischer Sicht mehrere Klassen von Operatoren gibt.

1. Als erstes werden die Rechenzeichen verarbeitet, die in Zahlen und Operatoren der zweiten und dritten Klasse umgewandelt werden können. Diese Rechenzeichen sind  $\%$  (= /100),  $\%$  (= /1000),  $\frac{1}{4}$  (= 0,25),  $\frac{1}{2}$  (= 0,5),  $\frac{3}{4}$  (= 0,75),  $^1$  (= ^1) und  $^\circ$  (= 1).
2. Die Rechenzeichen der dritten Klasse stehen hinter ihrem Parameter. Sie werden als zweites umgewandelt, da sie vor anderen Operatoren Vorrang haben. Diese Operatoren sind  $^2$  (= Sqr),  $^3$  (= Cube) und  $!$  (= Fakultät).
3. Die Rechenzeichen der zweiten Klasse verlangen zwei Parameter und stehen je dazwischen. Die Operatoren  $^$  (= Potenz),  $/$  (= Div),  $*$  (= Mul),  $-$  (= Sub),  $+$  (= Add) und  $\pm$  (= PlusMinus) werden in exakt dieser Reihenfolge verarbeitet, da dies ihrer Rangfolge entspricht.  $+$  und  $-$ , die ebenfalls als Vorzeichen vorkommen können, werden als solches gewertet wenn sie nicht hinter einer Zahl, Konstante oder einem Parameter stehen. Parameter dieser Funktionen

werden durch das Trennzeichen „;“ von einander getrennt, um sie beim eigentlichen Parsing besser unterscheiden zu können.

4. Das Rechenzeichen  $|Parameter|$  für Betrag, das seinen Parameter einschließt bildet einen Sonderfall und wird zuletzt umgewandelt.

Aus  $2 * \text{Pi} * \text{Wurzel} ( J / (m * g * a) )$   
 wird also `Mul(2; Mul(Pi; Wurzel(Divi(J; Mul(m; Mul(g; a))))))`

### 5 3 Das eigentliche Parsing

---

Nachdem der Parserstring generiert wurde, soll er nun „ausgerechnet“ werden. Ebenfalls wird jetzt geklärt, wie die Parameter im Formelkörper identifiziert werden.

Beim eigentlichen Parsing wird unterschieden zwischen **Operatoren**, **Nicht-Operatoren**, und **Trennzeichen**.

`Mul(2; Mul(Pi; Wurzel(Divi(J; Mul(m; Mul(g; a))))))`

**Trennzeichen** werden daran erkannt, dass sie den Zeichen „(“, „;“ oder „)“ entsprechen.

Einen **Operator** erkennt man daran, der entsprechende Teilstring jetzt in der *Globalfunctiontable* vorkommt.

Alles, was weder Operator noch Trennzeichen ist, ist ein **Nicht-Operatoren** und wird unterteilt in Variablen, Zahlen und Konstanten. Zahlen lassen sich in den Typ *Extended* (siehe [Kapitel fünf](#)) umwandeln. Die Teilstrings der Konstanten finden sich in der *Globalconsttable* wieder, einer globalen Tabelle die für jeden Konstantenbezeichner den dazugehörigen Wert enthält. Diejenigen Nicht-Operatoren, welche weder Zahlen noch Konstanten sind, müssen Variablen und somit Parameter sein.

Das Ergebnis jeder berechneten Formel wird auf einem internen Speicher des Parsers (*Variablen-Stack*) für spätere Verwendung zwischengelagert.

### 5 4 Das Ausrechnen verketteter Formeln

---

Um für größere Probleme mit dem Formelprogramm Lösungen finden zu können, müssen auch verkettete Formeln ausgerechnet werden können. Was unter verketteten Formeln zu verstehen ist, und wie diese ausgerechnet werden können, möchte ich an dieser Stelle erläutern.

Der Benutzer hat die Größen **a** und **b** gegeben und will **d** berechnen.

Das Formelprogramm kennt die Formeln

$$\begin{array}{l} \text{I. } c = a + b \quad \text{und} \\ \text{II. } d = c / (a * b) \quad . \end{array}$$

Das Programm soll jetzt in der Lage sein, **d** zu berechnen. Um die gesuchte Größe zu finden, ist es notwendig, zwei Formeln zu lösen. Dies übernimmt wiederum der integrierte Parser, welcher über eine Art *Variablen-Stack*, also einen Speicher für Variablen verfügt.

Zuerst werden **a** und **b** auf dem *Variablen-Stack* abgelegt, da ihre Werte bekannt sind. Soll der Parser die Formel I lösen, so wird er die beiden Variablen **a** und **b** finden. Deren Werte werden nun vom *Variablen-Stack* abgelesen und eingesetzt. Somit wird Formel I ausgerechnet und das Ergebnis auf dem *Variablen-Stack* abgelegt.

Dabei muss man beachten, dass der Parser lediglich Formelobjekte verarbeitet, wie sie aus [Kapitel 2.1](#) bekannt sind. Jedes solche Objekt verfügt gemäß [Kapitel 2.2](#) über einen Titel.

Betrachtet man die Formeln, so erkennt man, dass der **Titel der Formel** I dem **Variablenamen** in Formel II entspricht. Dies trifft auf alle mir bekannten Formelsysteme zu. So wird zum Beispiel in der Physik für jede Größe (meist) nur eine Bezeichnung verwendet.

Danach kann der Parser die Formel II lösen, da ihm die Variable **c** nun bekannt ist.

Auf diese Weise werden aus gegebenen Parametern Variablen, die anderen Formeln als Parameter dienen können. So ist es möglich, beliebig viele verkettete Formeln auszurechnen, deren Anzahl einzige und allein durch die Schranken der heutigen Hardware (also den Bestandteilen eines PC) eingegrenzt ist.

## 6 Wie stellt man die mathematischen Grundfunktionen numerisch stabil dar?

---

Um in die Formeln Werte einsetzen zu können und sie zu lösen, ist es notwendig, die mathematischen Grundfunktionen auf dem Rechner darzustellen. Die mathematischen Koprozessoren des Typs 80x87 bieten dafür folgende Operationen: Addieren, Subtrahieren, Multiplizieren, Sinus, Kosinus, Arcustangens,  $e^x$ , Trennung von Exponent und Matisse und einige weitere. Bei der Ausführung dieser Operationen können Fehler auftreten. Bei der Division zum Beispiel sind zwei verschiedene Ausnahmefälle, sogenannte „Exceptions“ möglich.

1. Division durch Null  
Ist der Divisor gleich Null, so wird die Exception „Division by zero“ ausgelöst.
2. Gleitkommaüberlauf  
Da die 80x87er Koprozessoren nur reelle Zahlen zwischen etwa  $-1,15 * 10^{4932}$  und  $1,15 * 10^{4932}$  darstellen können, ist es möglich, dass Ergebnisse entstehen, die den erfassbaren Bereich übersteigen, zum Beispiel  $1,15 * 10^{4932} / 0,5 = 2,3 * 10^{4932}$ . Bei solchen Berechnungen wird eine „Overflow“ Exception ausgelöst.

Dies kann zu falschen Ergebnissen führen, die vermieden werden müssen.

### 6 1 Das Abfangen von Exceptions

---

Exceptions können zum Einen zu extremer Verzerrung der Lösungsmenge oder sogar zum Absturz des ganzen Betriebssystems führen. Zum Anderen kann mit ihnen der korrekte Ablauf mathematischer Berechnungen geprüft werden. Deshalb soll sich dieser Abschnitt mit der Anwendung von Exceptions beschäftigen.

Wie mit Exceptions umgegangen werden kann, verdeutlicht dieses Stück Programmcode.

```
Function Add(X, Y: Extended): Extended;  
Begin  
  Try  
    Result := X + Y;  
  Except  
    Showmessage('Überlauf');  
    Result := 0;  
  End;  
End;
```

Die Funktion „Add“ soll die Summe der beiden reellen Zahlen X und Y als Ergebnis (Result) liefern. Mit dem reservierten Wort „Try“ wird dem Compiler mitgeteilt, dass, wenn eine Exception durch die folgenden Anweisungen ausgelöst wird, der Block zwischen „Except“ und „End;“ ausgeführt werden soll und andernfalls nicht. Verläuft die Berechnung normal, so enthält „Result“ das Ergebnis der Addition X + Y.

Wenn aber beispielsweise  $X = -1,15 * 10^{4932}$  und  $Y = -1,15 * 10^{4932}$  gilt, wird eine Exception ausgelöst. Denn die Funktion hätte  $-2,3 * 10^{4932}$  als Ergebnis, was aber zu groß ist, um vom 80x87 Koprozessor erfasst zu werden. Darum wird zuerst eine Nachricht auf dem Bildschirm („Überlauf“) ausgegeben. Danach wird das Ergebnis auf Null gesetzt.

Ohne diese Sperre würde das Programm abstürzen.

### 6 2 Die erweiterten Zustände – der Datentyp tRItem

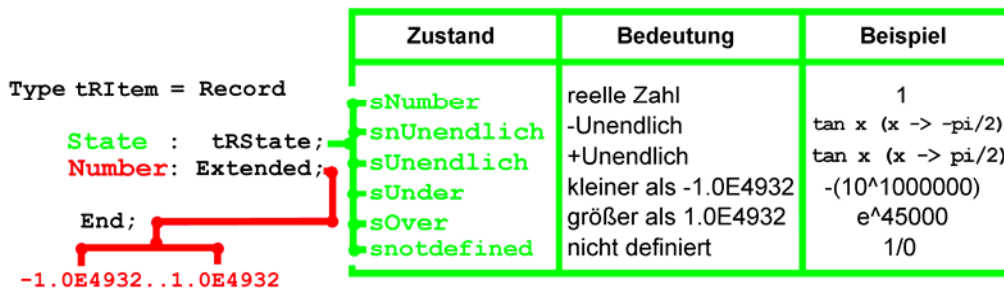
---

Nun ist das Programm in der Lage, fehlerhafte Operationen abzufangen. Jetzt soll dies auch genutzt werden. Nur auf diese Art wird die Mathematik des Programms wirklich stabil.

Dazu ist es logisch, die Zustände, die eine Zahl annehmen kann, zu erweitern.

Der neue Datentyp tRItem repräsentiert eine reelle Zahl, die sechs Zustände annehmen kann. Der aktuelle Zustand ist im Feld „State“ gespeichert.

Im Zustand sNumber entspricht eine Variable dieses Typs einer exakt definierten Zahl. Diese Zahl liegt zwischen  $-(10^{4932})$  und  $10^{4932}$ . Die Variable kommt  $-\infty$  gleich, wenn sie als snUnendlich gekennzeichnet ist. Wurde das Feld State mit sUnendlich belegt, so besitzt sie den Wert  $\infty$ . Ist die Zahl kleiner als  $-(10^{4932})$  so hat sie den Zustand sUnder inne und wenn sie größer als  $10^{4932}$  ist die Kennzeichnung sOver. Im Zustand snotdefined ist sie nicht definiert und kann jeden möglichen Wert zwischen inklusive  $-\infty$  und  $\infty$  annehmen.



An dieser Stelle ist es bereits gelungen, die mathematischen Grundfunktionen numerisch stabil darzustellen. Alle handelsüblichen Berechnungsprogramme, wie zum Beispiel Microsoft Excel oder Lotus 1 2 3 sind allerdings auch mehr oder weniger gut dazu in der Lage. Doch diese Programme liefern für jede Aufgabe nur ein einziges Ergebnis.

## 7 Wie schafft man es, alle möglichen Lösungen zu beachten?

Formula beachtet alle möglichen Ergebnisse. Dazu gehören Mehrdeutigkeiten bei Umkehrfunktionen, zum Beispiel bei der Wurzelfunktion, aber auch deren weiteres Verarbeiten. Im siebenden Kapitel soll also deutlich werden, wie dieser Aspekt von Formula wirkt und wie er entwickelt wurde.

Um es verständlicher zu machen, kann man Excel und Formula zwei Beispiele berechnen lassen. Zum einen die Lösung der Gleichung  $x^2 = 4$  und zum anderen den Term  $x = \tan \pi/2$ .

Formel	Microsoft Excel 97	Formula (im RAD-Modus, mehrere Lösungen aktiviert)
X = Wurzel (4)	= WURZEL(4) = 2	Wurzel(4) = (2 Möglichkeiten) = -2 oder = 2.  Die Formel enthielt eine Wurzel einer positiven reellen Zahl, die theoretisch auch den negativen Gegenpart zum positiven Ergebnis zurückliefern würde (z.B. Wurzel(4) = 2 und -2 da auch $(-2)^2 = 4$ gilt).
X = Tan( $\pi/2$ )	= TAN(PI()/2) = 1,63246E16	Tan(Pi/2) = (2 Möglichkeiten) = -Unendlich oder = Unendlich.  Es wurde ein Tangens mit einem ungültigen Argument aufgerufen, z.B. Tan(Pi / 2) im RAD-Modus, das Ergebnis wurde auf Unendlich und -Unendlich erweitert.

Excel berechnet beim ersten Beispiel korrekt zwei als Ergebnis. Jedoch ist es auch möglich, dass minus zwei eine Lösung ist. Der Tangens von  $\pi/2$ , also  $(\sin \pi/2) / \cos \pi/2$ , ergibt 1/0. Hier kalkuliert Excel falsch, es gibt lediglich eine sehr große Zahl aus. Setzt man diese in die Arcustangensfunktion ein, so ergibt sich annähernd  $\pi/2$ . Eigentlich ist dieser Term nämlich nicht definiert. Betrachtet man jedoch die Funktion genauer, so erkennt man, dass sie als Grenzwert  $\lim_{x \rightarrow \pi/2} \tan x$  bei  $x < \pi/2$  unendlich und bei  $x > \pi/2$

minus unendlich besitzt. Setzt man die Werte in die Arcustangensfunktion ein, so erhält man beide Male exakt  $\pi/2$ . Formula erkennt dies und liefert also  $-\infty$  und  $\infty$  als Lösung, sofern der Nutzer mehrere Ergebnismöglichkeiten zulässt.

Tut er dies nicht, so wird „nicht definiert“ zurückgegeben. Dieser Ausdruck ist zusätzlich kurz erklärt. Ein weiteres Merkmal der Mathematik von Formula ist es, Lösungen zu erklären, so dass selbst Laien in die Lage versetzt werden, auch kompliziertere Ergebnisse zu verstehen.

Ich habe bereits mit einigen Kalkulationsprogrammen gearbeitet, mich im Internet erkundigt und Herrn Kunert befragt. Dennoch sind mir bis jetzt keine Programme bekannt, die in der Lage sind, Gleichungen mit mehreren Ergebnissen richtig auszurechnen.

## 7 1 Verwalten mehrerer Lösungen

Um mehrer Lösungen zu ermöglichen, muss zuerst ein Konzept für deren Verwaltung entwickelt werden. Der Grundgedanke ist, Lösungen als Zahlenkolonnen darzustellen, die als neue Zahlentypen in Erscheinung treten.

Jeder Rechenschritt wird von Formula parallel für jede Ergebnismöglichkeit ausgeführt. Danach werden alle seine Ergebnisse zu einer Zahlenkolonne zusammengefasst. Dafür reicht der Zahlentyp `tRItem` nicht mehr aus.

Was bisher der Typ `tRItem` war, wird nun zu einem Eintrag in einer Liste, die der neue Typ `R` verwaltet.

Mit Hilfe der Methode `GetNumber` wird ein Eintrag aus der Liste gelesen. Die Methode `SetNumber` schreibt einen Eintrag in die Liste. Auch ist es nicht zweckmäßig, `R` als Record zu definieren.

Da nun eine Vielzahl von Methoden benötigt wird, um die Zahlen zu verwalten, ist es weitaus effizienter, `R` als Objekt (Class) zu programmieren.

Jetzt ist das Programm in der Lage, Zahlenkolonnen zu verwalten. Fällt in einer Berechnung mehr als eine Lösung an, so wird jede neue Lösung zuerst mit allen in der Kolonne verglichen. Wenn sie noch nicht vorkommt, so wird sie in die Kolonne eingefügt.

Würde auf das Vergleichen verzichtet, so könnte der Benutzer Ergebnisse wie

$x + y$  (2 Möglichkeiten)  
 $= 1$  oder  
 $= 1.$ 
erhalten.

Nach jedem größeren Rechenschritt werden die Einträge in der Kolonne der Größe nach sortiert, um später dem Benutzer eine bessere Übersicht zu gewährleisten. Da nicht definierte Elemente keine Größe besitzen, werden sie der Einfachheit halber nach dem größten Eintrag der Zahlenkolonne positioniert.

```

Type tRItem = Record
  State : tRState;
  Number: Extended;
End;
-1.0E4932..1.0E4932

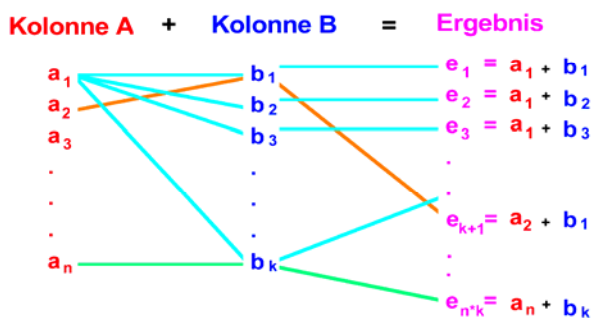
Type R = Class (tObject)
  Private
    fNumbers: Array of tRItem;
  Public
    Property Numbers[Index: Integer]: tRItem;
      Read GetNumber;
      Write SetNumber;
  End;
  
```

## 7 2 Rechnen mit mehreren Lösungen

Nachdem das Problem der Organisation mehrere Ergebnisse gelöst ist, stellt sich eine weitere Aufgabe: Wie rechnet man mit solchen Zahlenkolonnen?

### Addition von Zahlenkolonnen

als Beispiel für Operationen mit zwei Argumenten



Bei Operationen mit nur einem Argument ist dies ziemlich simpel: Zuerst wird eine leere Zahlenkolonne als Ergebnis erstellt. Dann wird das Ergebnis der Operation für alle einzelnen Teile der alten Zahlenkolonne in die neue eingetragen.

Die natürliche Zahl  $n$  gibt an, wie viele Einträge eine Zahlenkolonne besitzt.

Die Anzahl der Ergebnisse ( $A$ ) beträgt demnach  $A = n$ .

Bei Operationen mit zwei oder mehr Argumenten muss jeder Eintrag aus jeder Zahlenfolge mit jedem anderen Eintrag aus jeder anderen Zahlenfolge verrechnet werden.

Dies bedeutet Operationen mit  $x$  Argumenten haben  $A = \prod_{i=1}^x n_i$  Ergebnisse.

Die beiden Gleichungen für  $A$  treffen jedoch nur bei einem Spezialfall zu. Zwischen den Argumenten darf keinerlei Verbindung aus früheren Berechnungen bestehen. Auch dürfen sich keine identischen Lösungen ergeben, da keine Lösung zweimal in einer Zahlenkolonne vorkommen kann.

Obwohl die Gleichung für die Ergebnisanzahl  $A$  beim Produkt der Zahlenkolonnen  $Z_1(2; 3) * Z_2(3; 2; 4)$   $A = 2 * 3 = 6$  lautet, hat dieses fünf statt sechs Ergebnisse. Diese lauten  $E(4; 6; 8; 9; 12)$ . In beiden Zahlenkolonnen sind die Zahlen 2 und 3 enthalten. Da  $2 * 3$  und  $3 * 2$  identisch sind, und kein Eintrag in einer Zahlenkolonne doppelt vorkommen kann, fällt ein Ergebnis weg.

Auch stellt sich das Problem der Verbindung von Zahlenreihen, welches man sehr gut am Beispiel einer Multiplikation erklären kann. Multipliziert man zwei Zahlenfolgen  $Z_1$  und  $Z_2$ , so erhält man das Ergebnis  $E$ .

$$Z_1(2; 3) * Z_2(2; 3) = E(4; 6; 9)$$

Im Beispiel sind die Zahlenfolgen  $Z_1$  und  $Z_2$  gleich, sie besitzen die gleichen Einträge. Was passiert jedoch wenn die beiden Zahlenfolgen nicht nur gleich sind, sondern auch die selben?

$$Z_1(2; 3) * Z_1(2; 3) = E(4; 9)$$

In diesem Fall kann das Ergebnis nur zwei Einträge aufweisen, nämlich  $Z_{11} * Z_{11}$  und  $Z_{12} * Z_{12}$ . Warum das so ist wird schnell klar. Jeder Eintrag in einer Zahlenfolge stellt eine Ergebnismöglichkeit dar. Man kann die erste Gleichung also auch so aufschreiben:

$$x * y = E$$

Jeder Eintrag in  $Z_1$  stellt einen möglichen Wert für  $x$  und jeder Eintrag in  $Z_2$  einen möglichen Wert für  $y$  dar. Die zweite Gleichung müsste jedoch lauten:

$$x * x = E.$$

Bei der zweiten Gleichung stellt jeder Eintrag in  $Z_1$  einen möglichen Wert für  $x$  dar, und da  $x$  niemals gleichzeitig 2 und 3 sein kann, fällt 6 als Lösung weg.

Es ist also notwendig, vor jedem Rechenschritt, den Zusammenhang der Argumente zu überprüfen, denn nur so können kompliziertere Berechnung korrekt ausgeführt und alle, aber auch nur die richtigen, Ergebnisse angezeigt werden.

Wie in dem Beispiel rechts deutlich wird, ist es notwendig, nicht nur die Zahlenkolonnen direkt zu vergleichen, sondern auch ihre Abstammung zu klären, um sinnvolle Ergebnisse zu garantieren.

Dies wird im Programm dadurch gewährleistet, dass jede Zahlenkolonnen Zeiger auf alle Zahlenkolonnen enthält, aus denen sie errechnet wurde.

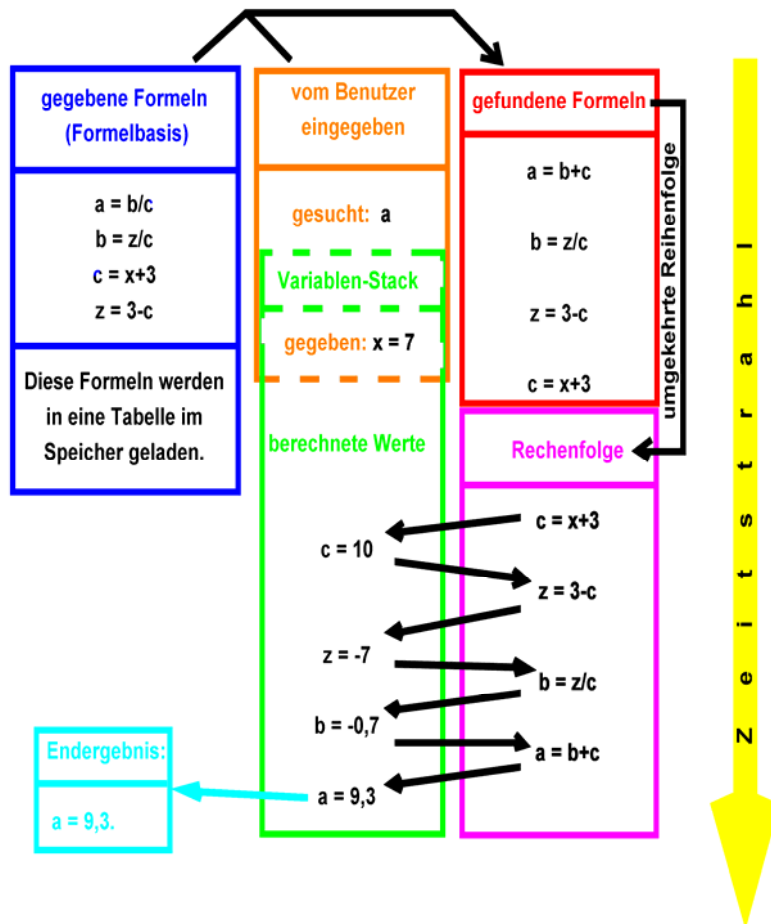
Auch jeder Eintrag einer Zahlenkolonne verfügt über ebensolche Zeiger auf seine Familie. So wird vor jeder Operation zuerst die Verwandtschaft der Zahlenkolonnen überprüft. Ist diese gegeben, so werden in der eigentlichen Berechnung nur diejenigen Einträge der Kolonne betrachtet, die an irgendeiner Stelle über den selben Vorfahr verfügen. In dem Beispiel wurden die verwandtschaftlichen Beziehungen der Einträge farblich hervorgehoben, die  $\color{blue}{6}$  im Ergebnis entsteht aus beiden „Stämmen“  $(-1 - -5$  und  $5 + -1)$ .

Die Überprüfung der Verwandtschaft funktioniert etwa so: Eine Zahlenkolonne bekommt den „Auftrag“ zu überprüfen, ob sie mit einer anderen Zahlenkolonne verwandt ist. Zuerst vergleicht sie, ob sie selbst diese Zahlenkolonne ist. Wenn nicht, so gibt sie den Auftrag nacheinander an alle ihre „Ahnen“ weiter. Und die wiederum an ihre, bis zur ersten Zahl in der Berechnung. Sobald ein Bezug festgestellt wurde, wird der Auftrag abgebrochen und „Wahr“ zurückgegeben, anderenfalls „Falsch“. Besteht

zwischen den Kolonnen eine Verwandtschaft, so muss auch zwischen deren Einträgen eine solche bestehen. Diese vergleichen sich auf die selbe Art. Besteht zwischen den Kolonnen keine Verwandtschaft, so kann sie auch zwischen deren Einträgen nicht bestehen. Da diese sich deshalb nicht darauf testen müssen, wird viel Rechenzeit gespart.

### 7 3 Ausrechnen

Bis zu diesem Kapitel beschäftigte sich der theoretische Teil meiner Besonderen Lernleistung lediglich mit den einzelnen Komponenten des Formelprogramms. Bevor jedoch die Bedienoberfläche erläutert wird, halte ich es für sinnvoll, das Zusammenspiel der funktionellen Bestandteile meines Programms zu beschreiben.



Wie in den vorherigen Kapiteln beschrieben, wird aus den vorhandenen Formeln mit Hilfe der Aufgabenstellung die richtige **Formelsequenz** gefunden. Die in ihr enthaltenen Formeln müssen nur noch ausgerechnet werden. Die zuletzt gefundene Formel ist immer diejenige, welche sich aus den gegebenen Parametern berechnen lässt. Ihr Ergebnis wird wiederum zum Ausrechnen der Formel, welche davor gefunden wurde, benötigt. Dies setzt sich bis zur ersten Formel fort, welche dann die Lösung der Aufgabe liefert. Die Formeln werden also in **umgekehrter Reihenfolge ihrer Auffindung berechnet**. Hierfür werden sie nacheinander dem Parser übergeben, welcher zum Lösen die interne Mathematik verwendet und die Ergebnisse vom Typ R im **Variablen-Stack** speichert. Nach jedem Rechenschritt müssen die Ergebnisse und deren Erklärung sowie eventuelle Fehler ausgegeben werden. Hat der Parser die erste Formel in der Liste berechnet, so wurde für die Aufgabe eine **Lösung** gefunden.

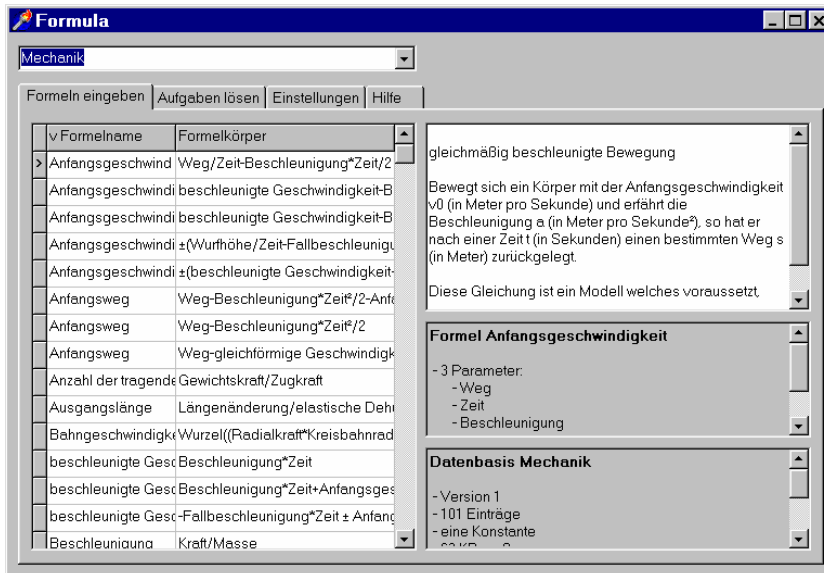
Da der Funktionsumfang des Formelprogramms jetzt komplett ist, fehlt nur noch eine Bedienoberfläche. Diese sollte einfach zu benutzen sein, über eine Hilfefunktion verfügen und auch graphisch ansprechend wirken.

Der Übersicht halber ist es vorteilhaft, auf große Menüs, die den Benutzer verwirren könnten, zu verzichten und mit möglichst wenig verschiedenen Schaltelementen einen möglichst hohen Bedienkomfort zu erreichen.

Nach diesen Gesichtspunkten ist eine Oberfläche mit vier Registern entstanden.

Die Oberfläche besitzt auf allen Computern unter dem Betriebssystem Windows etwa das selbe Aussehen, da sie sich automatisch an die Bildschirmauflösung anpasst. Dies wird durch Skalierung erreicht. Ändert man die Größe der Oberfläche, was wie bei den meisten Windows-Programmen möglich ist, so passen sich alle Bedienelemente automatisch an die neue Größe an, weshalb die Verhältnisse ihrer Abmessungen stets gleich bleiben.

Über allen Registern befindet sich links eine Liste, in der alle Formelbasen im aktuellen Verzeichnis aufgeführt sind. Durch betätigen des Knopfes daneben (mit den kleinen Pfeil nach unten) klappt diese Liste auf. Neue Formelbasen werden erstellt, indem einfach deren Name eingetippt wird.



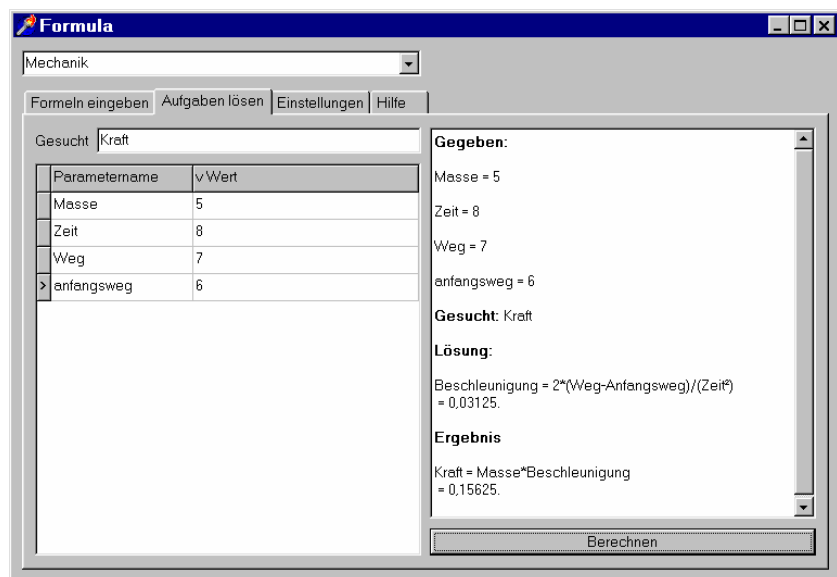
Das erste Fenster, welches zum Eingeben von Formeln dient, verfügt über zwei Bedienelemente und zwei Informationselemente.

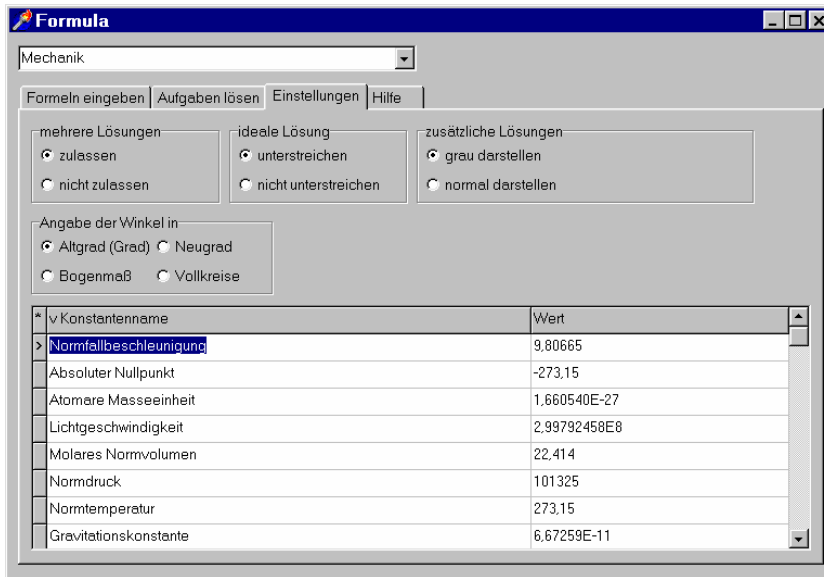
In der Tabelle links können Formeln eingegeben und bearbeitet sowie gelöscht werden. Änderungen werden umgehend auf die Formelbasis übertragen. In dem weißen Memo-Feld rechts oben werden die Kommentare zu den Formeln eingetragen und die beiden grauen Feldern unten erhalten Informationen über die ausgewählte Formel und über die geöffnete Formelbasis.

Der Register zum Lösen von Aufgaben verfügt wieder über zwei, vom Benutzer manipulierbare, Bedienelemente, welche sich auf seiner linken Seite befinden.

In der Zeile „Gesucht“ wird der Name der gesuchten Größe und in der Tabelle darunter je die Namen und Werte der bekannten Größen eingegeben.

Drückt man den Berechnen-Knopf rechts unten so wird die Lösung des Problems im Feld darüber angezeigt. Mit Hilfe eines PopUp-Menüs, welches aufklappt, wenn dieses Fenster mit der rechten Maustaste angeklickt wird, kann die Lösung in Dateien gespeichert, gedruckt oder in die Zwischenablage kopiert werden.



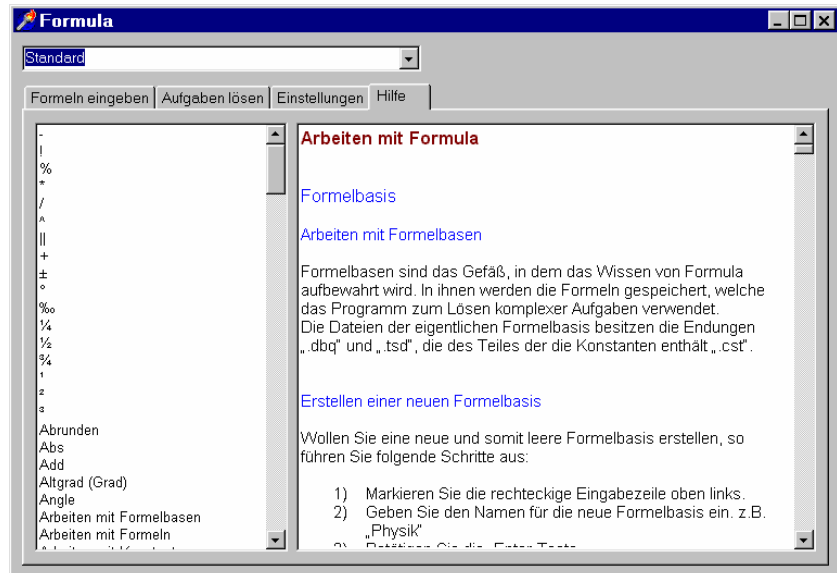


Im Register Einstellungen können die Optionen zu den Berechnungen geändert werden. Hier wird festgelegt, ob ein Term mehrere Lösungen haben kann oder nicht, und wenn ja ob die ideale Lösung markiert werden soll. Auch kann bestimmt werden, ob die Argumente von Winkelfunktionen in Altgrad, Neugrad, Bogenmaß oder in Vollkreisen angegeben werden. In der Tabelle unten werden Konstanten und ihre Werte für die aktuelle Datenbasis eingegeben.

Im vierten und letzten Register erhält der Benutzer Hilfe zum Gebrauch von Formula.

In der linken Liste sind die Stichwörter in alphabetischer Reihenfolge aufgelistet. Markiert man eines davon so wird im rechten Textfeld der entsprechende Eintrag angezeigt.

Auch kann man im rechten Memofeld das digitale Bedienhandbuch von Anfang bis Ende lesen, da dieses in logische Kapitel untergliedert ist und den Benutzer von den ersten Schritten bis zum vollen Gebrauch des Formelprogramms anleitet. Ebenso ist hier eine komplette Liste aller implementierten Funktionen von Formula enthalten.



Mit dieser Oberfläche ist das Formelprogramm fertig.

Jetzt muss es nur noch einige Zeit getestet werden um eventuelle, versteckte Fehler auszubessern und hier und da etwas Code zu optimieren.

Im Laufe des nunmehr fast drei Jahre andauernden Arbeitsprozesses an Formula habe ich mehr als siebenmal alles von Grund auf neu programmiert. Zweimal habe ich das Programm in Turbo Pascal 7.0 erstellt, bis eine Version gut genug erschien um diese Arbeit als Besondere Lernleistung anzumelden. Etwa aller acht Monate erweiterte sich meine programmiertechnische Kenntnis so weit, das alle bis dahin kreierte Programme veralteten und ich den Drang verspürte, von vorne zu beginnen.

Der Umstieg von Pascal auf Delphi 3.0 (und dann fast sofort auf Delphi 4.0) Mitte der elften Klasse führte dazu, dass ich die Windows-Programmierung von Grund auf und ohne große Vorkenntnisse erlernen musste. Dabei entstanden einige neue Formula-Versionen, die leider nur bedingt funktionstüchtig, geschweige denn graphisch-ästhetisch, waren. Schließlich veralteten sie, und es dauerte wieder ein halbes Jahr, bis ein effektives Formelprogramm entstand. Jedoch war auch dieses fehlerhaft, und es waren einige weitere Anläufe notwendig, um die jetzige Version zu erstellen.

Und dennoch gibt es einige Verbesserungsmöglichkeiten.

Formeln können zum Beispiel nicht nach einem ihrer Parameter umgestellt werden. Der dafür notwendige Aufwand überstieg leider die mir zur Verfügung stehende Zeit bei weitem. Zwar klingt diese Aufgabe nicht besonders schwer, denn ein Term wie „ $a = b+c$ “ lässt sich, auch rechentechnisch umsetzbar, schnell zu „ $b = a-c$ “ umstellen. Anders sieht es jedoch bei schwierigeren Formeln wie „ $a = x * \sin x$ “ aus. Und eine Erweiterung einzubauen, die nur bei einigen Formeln funktioniert, ist zwecklos. Weiterhin wäre es nicht schlecht, wenn das Programm auch mit imaginären Zahlen, Vektoren und Matrizen rechnen könnte. Dafür müsste auch die Oberfläche verändert werden.

Weiterhin wäre es nicht schlecht, wenn es eine Möglichkeit gäbe, die Einheiten physikalischer Größen in die Berechnung einfließen zu lassen.

Dennoch denke ich, mit Formula, ein kleines und nützliches Werkzeug entwickelt zu haben. Es kann weder durch Excel, Lotos, Maple, Mathematica oder Derive ersetzt werden, da all diese Programme andere Tätigkeitsfelder beanspruchen.

Weiterhin möchte ich anführen, dass ich keineswegs behaupte, Formula sei fehlerlos. Vor allem logische Fehler zeigen sich erst nach langer Benutzung, zu der ich leider bis jetzt keine Zeit hatte. Deshalb bin ich für jeden Hinweis dankbar, der es mir ermöglicht, mein Programm weiter zu verbessern.

Auch habe ich vor, im Laufe des nächsten Jahres eine zweite Version von Formula zu entwickeln.

Thomas Weise  
1999





## Anhänge

Den Nutzen von Formula kann man sehr gut an einer Übungsaufgabe beschreiben.

**Aufgabe 1:** Ein Körper der Dichte 4 Gramm pro Kubikzentimeter und einem Volumen von 700 Kubikzentimetern wird mit einer Kraft von 800 Newton 300 Meter weit gezogen. Wie groß ist die dabei verrichtete Leistung?

Um diese Aufgabe zu lösen, werden nur wenige Eingaben erforderlich.

Gesucht ist die Leistung.

Alle gegebenen Größen werden in der Tabelle darunter eingetragen.

Formeln eingeben		Aufgaben lösen		Einstellungen		Hilfe	
Gesucht <input type="text" value="Leistung"/>							
Parametername	v Wert						
Dichte	4						
Volumen	700						
> Kraft	800						
Weg	300						

mehrere Lösungen

zulassen

nicht zulassen

Da es sich um eine physikalische Aufgabe handelt, ist es nicht sinnvoll, mehrere Lösungsmöglichkeiten zu aktivieren.

Wird nun der Berechnen-Button gedrückt ...

Berechnen

**Gegeben:**

Dichte = 4

Volumen = 700

Kraft = 800

Weg = 300

**Gesucht:** Leistung

**Lösung:**

Masse = Volumen\*Dichte  
= 2800.

Beschleunigung = Kraft/Masse  
= 0,285714285714286.

Zeit = Wurzel(2\*Weg/Beschleunigung)  
= 45,8257569495584.

Arbeit = Kraft\*Weg  
= 240000.

**Ergebnis**

Leistung = Arbeit/Zeit  
= 5237,22936566382.

... erscheint im linken Lösungsfenster das Ergebnis.

Es ist druckfertig und enthält alle Zwischenergebnisse und Formeln.

Klickt man mit der rechten Taste der Maus in dieses Fenster, so öffnet sich ein PopUp-Menü.

Darin kann entschieden werden, ob das Ergebnis gespeichert (Als Text- oder Rich Text Format-Datei), gedruckt oder in die Zwischenablage kopiert werden soll.

In der nächsten Beispielaufgabe soll der Zweck der Möglichkeit, mehrere Lösungen anzeigen zu lassen, klar werden.

**Aufgabe 2:**

Eine Gondel mit der Masse 500 kg wird an ihrem Seil 10 Sekunden lang mit 3 Meter pro Quadratsekunde beschleunigt. Mit welchen Zugwinkeln kann die Kraft wirken, wenn die verrichtete Arbeit am Ende 2000 Newtonmeter betragen soll?

Wieder werden die gesuchten Größe und alle gegebenen Werte eingetragen.

Formeln eingeben		Aufgaben lösen		Einstellungen		Hilfe	
Gesucht <input type="text" value="Zugwinkel"/>							
	Parametername		v Wert				
	Masse		5				
	Zeit		10				
	> Beschleunigung		3				
	Arbeit		2000				

mehrere Lösungen

zulassen

nicht zulassen

Diesmal ist es jedoch sinnvoll, mehrere Lösungen zuzulassen. Diese Option würde ich bei allen Aufgaben empfehlen, die Berechnungen mit Winkeln enthalten könnten.

Nun folgt ein weiterer Mausklick auf das Berechnen-Button.

Berechnen

**Gegeben:**

Masse = 5

Zeit = 10

Beschleunigung = 3

Arbeit = 2000

**Gesucht:** Zugwinkel

**Lösung:**

Weg = Zeit\*Beschleunigung/2  
= 150.

Kraft = Beschleunigung\*Masse  
= 15.

**Ergebnis**

Zugwinkel = Arcuscossinus(Arbeit/(Kraft\*Weg))  
» 2 Möglichkeiten  
= 27,2660444507328 oder  
= 332,733955549267.

Die Formel enthält eine Arcuscossinus-Funktion, welche zwei Ergebnisse aufweist.

Die idealen Lösungen wurden unterstrichen.

Diesmal werden zwei Ergebnisse gefunden.

Die Gondel kann einmal mit einer Winde oberhalb des Seils und einmal unterhalb des selben gezogen werden.

Das Ergebnis wird kurz erklärt, und die naheliegendste Möglichkeit unterstrichen.

Einletztes Beispiel bietet die Berechnung Dreiecksseiten. Bei bestimmten Aufgabenstellungen ist es nämlich möglich, dass zwei Dreiecke als Ergebnis entstehen.

### Aufgabe 3:

Von einem Dreieck sind Längen der Seiten a (3 cm) und b (4 cm) bekannt. Wie lang ist die Seite c, wenn der Winkel Alpha  $34^\circ$  beträgt?

Wieder müssen die notwendigen Eingaben durchgeführt werden.

Formeln eingeben		Aufgaben lösen		Einstellungen		Hilfe	
Gesucht <input type="text" value="c"/>							
<input type="checkbox"/>	Parametername	Wert					
<input type="checkbox"/>	b	4					
<input type="checkbox"/>	a	3					
<input checked="" type="checkbox"/>	alpha	34					

mehrere Lösungen

zulassen

nicht zulassen

Da es sich, wie im letzten Beispiel, um eine Berechnung handelt, in der Winkelfunktionen benutzt werden könnten, sollten wieder mehrere Lösungen zugelassen werden.

Nach einem letzten Klick auf das Berechnen-Button...

Berechnen

**Gegeben:**

b = 4

a = 3

alpha = 34

**Gesucht:** c

**Lösung:**

beta = arcsin(b\*sin(alpha)/a)  
 » 2 Möglichkeiten  
 = 48,209846768194 oder  
 = 131,790153231806.

Die Formel enthält eine Arcussinus-Funktion, welche zwei Ergebnisse aufweist.

Die idealen Lösungen wurden unterstrichen.

gamma = 180-alpha-beta  
 » 2 Möglichkeiten  
 = 14,209846768194 oder  
 = 97,790153231806.

**Ergebnis**

c = a\*sin(gamma)/sin(alpha)  
 » 2 Möglichkeiten  
 = 1,316937258218 oder  
 = 5,3153633222233.

... wird die Lösung der Aufgabe angezeigt.

Jetzt kann man erkennen, dass zwei Dreiecke, mit den in der Aufgabe genannten Eigenschaften, existieren.

Demzufolge enthält die Ergebnismenge auch zwei unterschiedliche Längen für c.

Formula benötigte zur Lösung pro Beispiel unter zehn Sekunden Zeit. Ein Mensch hätte weit länger gebraucht.

Deshalb denke ich, dass mein Programm durchaus eine Daseinsberechtigung besitzt.



TS Formula

ALPHA Version

Users Manual

lizensiert für: Frau Dippmann

## Anforderungen an das System:

- Ein 286er Prozessor
- 80x87er mathematischer Koprozessor (ist standardmäßig in allen Modellen eingebaut)
- 600 KB freier Konventioneller Speicher
- 1 MB freier Festplattenspeicher
- 3½ Zoll Diskettenlaufwerk
- MS-DOS 4.0 bzw. Windows 2.0

## Inhalt:






1. Kapitel: Die Installation
2. Kapitel: Wie gebe ich eine Formel ein?
3. Kapitel: Wie geben ich eine Konstante ein?
4. Kapitel: Wie berechne ich ein Ergebnis?

Anmerkung: Die Alphaversion entstand vor etwa zwei Monaten und ist derzeit auch auf Grund neuer mathematischer Erkenntnisse (Schulunterricht) etwas überholt. Die neue Version soll unter Windows 95 / 98 entstehen, via Delphi 4.0 Standard, und wesentlich benutzerfreundlicher, intelligenter und schneller, genauer und, unter Umständen auch auf Komplexe Zahlen und Vektoren sowie auf Matrix –Berechnungen erweitert sein. Eine Komponente für Berechnungen im Bereich der reellen Zahlen ist derzeit in Arbeit und wird voraussichtlich in den nächsten Ferien fertig werden. Danach werde ich mich der Datenbankprogrammierung widmen, dann der neuen Lösungstechnik (Versuch: DLL während der Laufzeit erstellen → tausendmal mehr Tempo) und zuletzt einer schönen Oberfläche.

Tschaikowskistraße 30  
09130 Chemnitz  
Telefon: 4015607  
Telefax: 4015607  
Tutorgruppe Schmidt klasse 11/6  
E-Mail an der Schule: MHAUSER

## 1. Kapitel: Die Installation

In der Alphaversion gibt es noch kein Installationsprogramm, die Dateien müssen manuell kopiert werden. Da das Programm im MS-DOS-Modus am besten funktioniert, wird hier die Installation im MS-DOS-Modus erklärt.

1. Schritt: Auf das Laufwerk gehen, auf das TS Formula installiert werden soll. (z.B. C:\ )
2. Schritt: Das Verzeichnis erstellen, in das TS Formula installiert werden soll. (z.B. *md Formula* )
3. Schritt: *cd <Diskettenlaufwerksbezeichnung>*  (entweder A:\ oder B:\)
4. Schritt: *copy \*.\* <Laufwerk von 1.> <Verzeichnis von 2.>* 
5. Schritt: *copy Source <Laufwerk von 1.> <Verzeichnis von 2.> <Source>* 

### Anmerkungen

Der Ordner „Source“ enthält die Sourcecodes des Programms, diese sollten nicht weitergegeben werden. Die „.DOC“ Dateien sind Word '97 Dokumente und enthalten diese Handbücher. Das mitgelieferte Tool „Convert.EXE“ dient der Umrechnung in verschiedene Zahlensysteme und ist (neben dem Turbo Pascal Compiler „TPC.EXE“) das einzige Programm, was ich nicht alleine erschaffen haben. (Es entstand im „TP-Kurs“ in den Sommerferien mit Alexander Sävert, Mario Öttler und Sebastian Degenkolb).

Sofern Sie über Turbo Pascal 7.0 verfügen, können sie die Sourcecodes modifizieren. Da aber eine Unit „Objects“ heißt, ist es möglich, das diese beim compilieren die TP7-eigene Objects-Unit ausknockt und deshalb: für Schäden, die dadurch entstehen, übernehme ich keine Haftung. Generell dürften keine Schäden entstehen, da ich das Programm eigentlich sicher programmiert habe und auch umfassend testete.

Fehler:




Ist ein Fehler aufgetreten, erhalten sie kompetente Hilfe zu TS Formula telefonisch bzw. faxmäßig unter 4015607.

## 2. Kapitel: Wie gebe ich eine Formel ein?

War die Installation erfolgreich, so befindet sich jetzt in dem Ordner, den Sie für TS Formula ausgewählt haben, die „DataRead.EXE“.


Wie man nun Formeln eingibt erkläre ich an einem Beispiel:

Die Formel  $v = s/t$  ist einzugeben.

1. Starten des Programms „DataRead.EXE“ : *DataRead* 
2. Jetzt müßte die Meldung  
„Die Datenbank wurde korrekt initiiert.“  
in grüner Schrift auf dem Bildschirm erscheinen. (Erscheint sie nicht, lesen Sie in Kapitel 1, „Fehler“ weiter).  
Darunter steht nun in weißer Schrift  
„Bitte geben Sie Ihre Formel in diesem Format an:  
<Formelname> <Enter> <Formel> <Enter>“.
3. Der Name unserer Formel lautet „v“. (Warum ist hoffentlich klar).
4. Also gegeben Sie *v*  ein.
5. Ein „=“ erscheint automatisch am Bildschirm, das bedeutet, das TS Formula den Formelname anerkannt hat.
6. Als nächstes müssen wir die Formel angeben, bei uns wäre dies „Divi(s,t)“.  
Geben Sie also ein: *Divi(s,t)*   
An dieser Stelle möchte ich erklären, was es mit den vordefinierten Funktionen aus sich hat.

Will man zum Beispiel die beiden Zahlen x und y addieren, so ist es bei Formula nicht möglich, einfach  $x + y$  hinzuschreiben, man muß stattdessen `add(x,y)` eingeben. Will man die drei Zahlen a, b und c miteinander multiplizieren, so ist es nötig, folgendermaßen zu verfahren: `mul(a,mul(b,c))`.

Warum so kompliziert? Die Verwendung dieser Funktionen ist zwingend notwendig, um solche Nettigkeiten wie „Fließkommaüberlauf“ oder „Division by Zero“ zu vermeiden. Außerdem ermöglicht sie erst die Verwendung „erweiterter mathematischer Gesetzmöglichkeiten“. Sämtliche vordefinierten Funktionen sind im Heft „Die implementierten Funktionen und Konstanten“ verzeichnet.

7. Als nächstes erscheint die Meldung:  
„Nun geben Sie bitte die Parameter der Formel mit Leerzeichen getrennt ein!“  
Bei unserer Formel sind s und t die Parameter, geben Sie als ein: `s t`
8. Die Meldung „Die Formel wurde korrekt gespeichert“ erscheint, und zeigt uns, das die Formel nun dem Formelschatz des Programmes beigefügt wurde.
9. Jetzt werden wir gefragt: „Eine weitere Formel eingeben [jn] ?“. Bejahen Sie dies durch Druck auf , um noch etwas zu üben.
10. Wir landen wieder bei Schritt Nummer 2.


Nun probieren wir, die Formel „ $s = a/2 * t^2$ “ einzugeben.

Der Formelname lautet „s“, die Formel ist etwas komplizierter, sie lautet „`Mul(Divi(a,2),Sqr(t))`“.

(Jetzt ist eine gute Gelegenheit, um sich das Nachschlagewerk „Die implementierten Funktionen und Konstanten“ anzuschauen.)

Die Parameter der Funktion sind logischerweise „s t“.

Anmerkung: Formelnamen und Parameter können bis zu 30 Buchstaben lang sein und dürfen länderspezifische Sonderzeichen wie ä, ü, ß, @, \$, %, ! und viele andere (siehe „Objects.Pas“) enthalten.

Zu guter letzt geben wir noch die Formel „ $a = v/t$ “ ein. (Formelname: „a“; Formel: „`Divi(v,t)`“ und die Parameter sind „v t“). Ein und verneinen nun die obligatorische Frage „Eine weitere Formel eingeben [jn] ?“ durch .

Das Programm wünscht uns noch einen schönen Tag und mit einem letzten Druck auf  ist es beendet.

### 3. Kapitel: Wie gebe ich eine Konstante ein?

Ein Konstante wird genauso wie eine Formel eingegeben, nur das der „Formelkörper“ dem Wert der Konstanten entspricht und die Konstante keine Parameter hat.

Beispiel:  $g = 9.81$

Formelname: „g“

Formel: „9.81“


Parameter:

### 4. Kapitel: Wie berechne ich ein Ergebnis?

Bei der Alphaversion dient dem Berechnen von Ergebnissen das Programm „Formel.EXE“.

Diese Erklärung bezieht sich auf die der vorigen zwei Kapitel, stellen Sie also sicher, das Sie diese gelesen und auch bearbeitet haben.



Angenommen, man hat eine geradlinige gleichmäßig beschleunigte Bewegung, bei der nach 10s die Endgeschwindigkeit 20 m/s beträgt. Wir wollen wissen, wieviel Weg zurückgelegt wurde.

1. Starten Sie das Programm „Formel.EXE“ `<Formel>` 
2. Jetzt müßte die Meldung  
„Die Datenbank wurde korrekt initialisiert.“  
in grüner Schrift auf dem Bildschirm erscheinen. (Erscheint sie nicht, lesen Sie in Kapitel 1, „Fehler“

weiter).

Darunter steht nun in weißer Schrift

3. „Wollen Sie Parameter eingeben [jn] ?“

Bejahen Sie dies durch Druck auf  .



Nun können Sie lesen:

„Bitte geben Sie die bekannten Parameter in der Form

<Parametername> <Enter> <Wert> <Enter> ein!“

Was ist nun unser erster Parameter? Die Zeit  $t$  mit dem Wert 10.



Einheiten können bei TS Formula nicht mitgeführt werden.

Geben Sie also ein  $t$   ein. Es erscheint automatisch ein „=“, nun geben Sie ein 10  ein.

Als nächstes werden Sie gefragt „Weitere Parameter eingeben [jn] ?“.

Da wir zwei Parameter haben, drücken Sie .

Unser nächster Parameter ist die Geschwindigkeit  $v$  mit dem Wert 20.


Geben Sie also  $v$   ein. Wieder erscheint ein „=“, nun geben Sie 20  ein.

Diesmal verneinen Sie folgende Frage.

PS: Es können bis zu 220 Parameter eingegeben werden.

4. Die folgende Frage lautet „Bitte geben Sie jetzt die gesuchte Größe an: “

Und die ist bei uns der Weg  $s$ .

Geben Sie als  $s$   ein.


5. Was passiert nun?

Zunächst erscheint die Meldung „Das Ergebnis wird nun berechnet!“.

Es folgen zwei Zeilen, in denen Formula die Formeln anzeigt, die es aus seiner Datenbasis herausgesucht hat: „ $A = \text{DIVI}(V,T)$ “ und „ $S = \text{MUL}(\text{DIVI}(A,2),\text{SQR}(T))$ “.

Darauf wird die Meldung des Turbo Pascal Compilers („TPC.EXE“) sichtbar („Turbo Pascal Version 7.0 Copyright.....“).

Nach der folgenden Lehrzeile steht geschrieben „Das Ergebnis lautet 1.00000000E+002“, es wurden also 100m zurückgelegt.

6. Nach dem Druck auf  wird gefragt, ob man eine weitere Berechnung durchführen will, bejaht man dies kommt wieder bei Schritt 2 raus, anderenfalls hat man das Programm verlassen

### **Ist Ihnen an der Proberechnung etwas aufgefallen?**

Die für die bekannte Formel für den Weg benötigten Parameter waren nur zum Teil gegeben: der für die Beschleunigung  $a$  fehlte, stattdessen war die Endgeschwindigkeit  $v$  gegeben.

Das Programm hat selbstständig nach einer Formel gesucht, mit deren Hilfe aus den gegebenen Parametern  $a$  berechnet werden konnte.

Dies war die schon einmal angesprochene Suche um eine Ebene abwärts, die theoretisch bis in 180 Ebenen Tiefe fortgesetzt werden kann.

[Meine besondere Lernleistung](#)

[Die implementierten Funktionen](#)

# Protokoll

Datum	Beginn	Ende	Tätigkeit
12.10.1998			Alphaversion als DOS-Programm für Herrn Sachse. Dazu werden Anleitungen und Erklärungen verfaßt.
17.11.1998			Übertragung des PASCAL-Code mit Hilfe von Frank Stolle in Delphi und somit ins 16Bit Windows. Entscheidung, alles für das 32Bit Windows neu zu programmieren, da dort höhere Effizienz zu erwarten ist. Außerdem hätte das Programm einen höheren Nutzen für den Anwender hätte und zeitgemäßer wäre. Damit ist der Kauf von Borland Delphi verbunden sowie dessen autodidaktisches Erlernen.
31.01.1999	16:50		Schreiben von Texten zur Besonderen Lernleistung.
03.02.1999	15:00 16:45		Erstes Eintreffen in der Uni, Sekretär schickt mich zu Herrn Schulze, der um diese Zeit nicht mehr in der Uni ist, deshalb lasse ich mein Projekt da und komme morgen wieder. Schreiben eines Textes zur Besonderen Lernleistung.
04.02.1999	13:15 22:00	14:15	Zusammentreffen mit Herr Schulze, Herr Schulze schickt mich zu Herrn Kunert, erstes Zusammentreffen mit Herr Kunert, Herr Kunert willigt ein, mich zu betreuen. Erstellen einer kurzen Anmeldung meines Projektes.
06.02.1999	18:20		Schreiben einiger Texte zur Besonderen Lernleistung.
09.02.1999	09:00	11:00	Arbeiten am Datenbankobjekt und der Datenbankstruktur von meinen Datenbanken.
09.02.1999	13:00	17:00	
10.02.1999	09:00	11:00	
10.02.1999	13:00	17:00	
11.02.1999	09:00	11:00	
11.02.1999	13:00	17:00	
11.02.1999	18:30	21:00	
12.02.1999	13:00	14:00	
12.02.1999	17:00	21:00	Erstellen der genauen <a href="#">Beschreibung meiner Besonderen Lernleistung</a>
13.02.1999- 01.04.1999	16.30	21.00	Erarbeiten einer Windows-Version, welche auf Grund meiner noch geringen Fähigkeiten in der Delphi-Programmierung wenig effizient ist. Darum beschließe ich, noch einmal anzufangen und verwerfe diese Version. Mein bis jetzt erlernten Fähigkeiten werde mir helfen, die nächste Version entscheidend zu verbessern.
03.05.1999	22.00 16.00	21.45	E-Mail-Verkehr mit Dr. Kunert, um ein Treffen Betreff der implementierten Mathematik vorzubereiten. Arbeit an der Oberfläche und Datenbasis.
11.05.1999	18.00	21.00	Antwort von Dr. Kunert auf Mail vom 03.05.1999 erhalten. Weitere Arbeit an der Datenbasis.
14.05.1999- 16.05.1999	17:30	22:00	Arbeit an dem Teil der Oberfläche, der zur Ein- und Ausgabe der Formeln dient, um bald mit dem Lösungsalgorithmus beginnen zu können. Tests und Neuarbeitung von mathematischen Operationen.
17.05.1999- 19.05.1999	17:00	21:00	Fertigstellen einer Eingabemaske für Formeln, die halbwegs funktioniert und später nur noch leicht modifiziert werden muß.
20.05.1999	16:30	18:10	Konferenz mit Herr Kunert, um Möglichkeiten für Sonderfälle des Systems der reellen Zahlen zu klären. Erstellen einer Liste mit mathematischen Operationen und Diskussion über die weitere Verfahrensweise. Werde den Lösungsalgorithmus jetzt anfangen, um dann bald mit der Theorie zu beginnen.
21.05.1999- 23.05.1999	18:00	21:00	Verbessern kleinere Fehler in der Eingabemaske.
06.06.1999	18:30	22:15	Weitere Arbeit an Eingabemaske und dem Objekt zur

			Übersetzung von Formelcode in PASCAL-Code.
07.06.1999	17:30 22:30	21:45	Weiteres Verbessern der Oberfläche, Versuch, die heute bearbeiteten Module an Herrn Kunert zu mailen, welcher auf Grund der Dateigröße scheitert.
12.06.1999- 18.06.1999  19.06.1999	16:30	21:30	Arbeit an der Lösungsmaske, Anpassen der Eingabemaske und Verbesserung der Datenbasis für Zugriffen der Lösungsmaske und höhere Performance. Fortsetzung der Arbeit am Lösungsalgorithmus, dabei stoße ich auf ein schweres Problem der Eingabemaske: es können keine Formeln gleichen Titels eingegeben werden.
24.06.1999	09:00	13:00	Lösen dieses Problems nach langen Suchen. Jetzt kann der Lösungsalgorithmus weiter entwickelt und verbessert werden. Ich entwickle die Umwandlung von Formelcode in PASCAL-Code weiter, so daß zum Lösen jetzt fertige DLL's erstellt werden, welche vom Programm geladen und ausgeführt werden.
25.06.1999	09:00	22:00	Nun kann ich beginnen, die mathematischen Funktionen zu implementieren, wie Dr. Kunert sie mit mir besprochen hat (20.05.1999).
26.06.1999	17:00 10:00-17:00		Beginn des Schreibens eines Dokumentes, welches die implementierten mathematischen Funktionen erklärt, welche heute ebenfalls in Arbeit sind.
27.06.1999	12:00	22:00	Schreiben dieses Dokumentes und Arbeit an den implementierten Funktionen.
28.06.1999- 02.07.1999 03.07.1999	16:00  12:00	22:00  20:00	Arbeit an den implementierten mathematischen Funktionen, verbessern des Lösungsalgorithmus: Zwischenergebnisse werden angezeigt (über Callback-Funktion).
04.07.99	10:00	20:00	Arbeit an der Oberfläche: Idee mit den drei Fenstern. Überarbeiten der GreyMemo-Komponente.
05.07.99	15:00	22:00	Arbeit an der Oberfläche und der AimMap-Komponente.
06.07.99	16:00	22:00	Idee mit der parallelen Mathematik, erster Versuch scheitert.
07.07.99	15:12  15:38	15:37  22:00	Telefonkonferenz mit Herrn Kunert betreffs der Entwicklung paralleler Mathematik und deren Einsatzmöglichkeiten. Überarbeitung einiger Grundfunktionen. Entwicklung eines sinnvollen Abbruchalgorithmus.

Änderungen Vorbehalt, Thomas Weise, 27.06.1999

Die implementierten Funktionen  
und Konstanten

# Die implementierten Funktionen und Konstanten

Dieser Abschnitt enthält eine kurze Zusammenfassung der bereits vordefinierten Funktionen und Konstanten, deren Verwendung dem Benutzer offen steht.

Diese Zusammenfassung ist noch unvollständig, da sie noch keine Auskünfte über die Ausnahmefälle durch die „erweiterten mathematischen Gesetzmäßigkeiten“ enthalten. Diese habe ich noch nicht beigefügt, da sie die Größe des Dokuments verzehnfachen würden.

Angaben über die Darstellungsmöglichkeiten von Zahlen auf dem Computer beziehen sich einzig und allein auf die mathematischen Koprozessoren vom Type 80x87.

## Konstanten

Die folgenden Konstanten werden hauptsächlich für Rechnungsinterne Zwecke verwendet, ihre Verwendung steht jedoch dem Benutzer offen.

Name	Wert	Bedeutung
MaxR	$1.18973149535723176499999999 * 10^{4932}$	MaxR steht für die größte Zahl, die der Computer darstellen kann, bei Rechnungen steht sie für Unendlich ( $\infty$ ).
MinR	$-1.18973149535723176499999999 * 10^{4932}$	MinR steht für die kleinste Zahl, die der Computer darstellen kann, bei Rechnungen steht sie für $-\infty$ .
MinStep	$3.3621031431120935070 * 10^{-4932}$	Die kleinste positive reelle Zahl größer als 0, die der Computer darstellen kann.
Max1Add	$1 * 10^{18}$	Die kleinste positive reelle Zahl die der Computer zu 1.0 addieren kann, so das sich das Ergebnis noch ändert. Kleiner positive reelle Zahlen gehen bei der Addition bzw. Subtraktion verloren, da der Computer die Exponenten zuerst angleicht und die Zahlen so zu 0.0 werden.
Min1Add	$-1 * 10^{18}$	Die größte negative reelle Zahl die der Computer zu 1.0 addieren kann, so das sich das Ergebnis noch ändert. Die Erklärung ist analog zu Max1Add.
HalfMaxR	$\text{MaxR} / 2.0$	Die Hälfte der größten reellen Zahl, die der Computer darstellen kann, bei Rechnungen steht sie für $\frac{1}{2}$ Unendlich; ( $0.5 * \infty$ ).
HalfMinR	$\text{MinR} / 2.0$	Die Hälfte der kleinsten reellen Zahl, die der Computer darstellen kann, bei Rechnungen steht sie für $-\frac{1}{2}$ Unendlich ( $-0.5 * \infty$ ).
RootMaxR	$\sqrt{\text{MaxR}}$	Die Wurzel der größten reellen Zahl, die der Computer darstellen kann, bei Rechnungen steht sie für $\sqrt{\text{Unendlich}}$ ; ( $\sqrt{\infty}$ ).
RootMinR	$-\sqrt{\text{MaxR}}$	Der negative Gegenpart zur Wurzel der größten reellen Zahl, die der Computer darstellen kann, bei Rechnungen steht er für $-\sqrt{\text{Unendlich}}$ ; ( $-\sqrt{\infty}$ ).
LnMaxR	Natürlicher Logarithmus von MaxR	Der natürliche Logarithmus der größten reellen Zahl, die der Computer darstellen



Die folgenden Konstanten sind nur für den Benutzer geschaffen und dienen der Verwendung in physikalischen Gleichungen.

Name und Erklärung	Wert
Absoluter_Nullpunkt Der absolute Nullpunkt in °C angegeben.	-273.15
Atomare_Masseinheit Die atomare Masseinheit 1u angegeben in Kg.	$1.660540 * 10^{-27}$
LichtGeschwindigkeit Die Lichtgeschwindigkeit im Vakuum in m/s.	$2.99792458 * 10^8$
Molares_Normvolumen Das molare Normvolumen in l/mol.	22.414
NormDruck Der Normdruck in Pa.	101325
NormFallBeschleunigung Die Normfallbeschleunigung auf der Erden in m/s <sup>2</sup> .	9.80665
NormTemperatur Die Normtemperatur in K.	273.15
GravitationsKonstante Die Gravitationskonstante in m <sup>3</sup> */ (kg * s)	$6.67259 * 10^{-11}$
ElektrischeFeldkonstante Die elektrische Feldkonstante in A * s / (V * m)	$8.854188 * 10^{-12}$
MagnetischeFeldkonstante Die magnetische Feldkonstante in V * s / (A * m)	$1.256637 * 10^{-6}$
AVOGADROKonstante Die Avogadro-Zahl in 1/mol.	$6.022136 * 10^{23}$
BOLTZMANNKonstante Die Boltzmannkonstante in J/K.	$1.380658 * 10^{-23}$
HUBBLEKonstante Die Hubblekonstante in Km /s * Mpc)	55
PLANCKKonstante Das Plancksche Wirkungsquantum in J * s.	$6.626076 * 10^{-34}$
RYDBERGGKonstante Die Rydbergkonstante in 1/m	$1.09737315 * 10^7$
SolarKonstante Die Solarkonstante in W / m <sup>2</sup>	$1.359 * 10^3$
UniverselleGaskonstante Die universelle Gaskonstante in J / (K * mol)	8.314510
Elektron_Ladung Die Elementarladung in C	$1.602177 * 10^{-19}$
Elektron_RuheMasse Die Ruhemasse eines Elektreons in Kg.	$9.1093897 * 10^{-31}$
Elektron_SpezifischeLadung Die spezifische Ladung eines Elektrons in C/Kg	$1.75881962 * 10^{11}$
Neutron_Ruhemasse Die Ruhemasse eines Neutrons in Kg.	$1.6749286 * 10^{-27}$
Proton_Ruhemasse Die Ruhemasse eines Protons in Kg.	$1.6726231 * 10^{-27}$

## Funktionen

Die folgenden Funktionen können bei selbstdefinierten Funktionen verwendet werden.

Syntax	Bedeutung
<b>Grundfunktionen</b>	

Add (a <sub>1</sub> , a <sub>2</sub> )	a <sub>1</sub> + a <sub>2</sub>
Sub (a <sub>1</sub> , a <sub>2</sub> )	a <sub>1</sub> - a <sub>2</sub>
Mul (a <sub>1</sub> , a <sub>2</sub> )	a <sub>1</sub> * a <sub>2</sub>
Divi (a <sub>1</sub> , a <sub>2</sub> )	a <sub>1</sub> / a <sub>2</sub>
IDivi (a <sub>1</sub> , a <sub>2</sub> )	Int (a <sub>1</sub> / a <sub>2</sub> ) wie oft geht a <sub>2</sub> ganz in a <sub>1</sub>
Modulo (a <sub>1</sub> , a <sub>2</sub> )	a <sub>1</sub> - IDivi(a <sub>1</sub> , a <sub>2</sub> ) * a <sub>2</sub> (Rest der I-Division)
Sqr (a <sub>1</sub> )	a <sub>1</sub> <sup>2</sup>
Sqrt (a <sub>1</sub> )	√a <sub>1</sub>
Exp (a <sub>1</sub> )	e <sup>a<sub>1</sub></sup>
Ln (a <sub>1</sub> )	Natürlicher Logarithmus von a <sub>1</sub>
Expb (a <sub>1</sub> )	2 <sup>a<sub>1</sub></sup>
Lb (a <sub>1</sub> )	Binärer Logarithmus von a <sub>1</sub>
Potenz (a <sub>1</sub> , a <sub>2</sub> )	a <sub>1</sub> <sup>a<sub>2</sub></sup>
Log (a <sub>1</sub> , a <sub>2</sub> )	Logarithmus von a <sub>2</sub> zur Basis a <sub>1</sub>
ShiftR (a <sub>1</sub> , a <sub>2</sub> , a <sub>3</sub> )	a <sub>1</sub> wird zur Basis a <sub>2</sub> um a <sub>3</sub> nach rechts verschoben (beim binären Assemblerbefehl shr ax, n ist 2 die Basis, ax würde a <sub>1</sub> und n a <sub>3</sub> entsprechen.
ShiftL (a <sub>1</sub> , a <sub>2</sub> , a <sub>3</sub> )	a <sub>1</sub> wird zur Basis a <sub>2</sub> um a <sub>3</sub> nach links verschoben. Erklärung siehe ShiftR.
<b>comparative Funktionen</b>	
Round (a <sub>1</sub> )	rundet a <sub>1</sub>
HighRound (a <sub>1</sub> )	rundet a <sub>1</sub> immer auf
Sign (a <sub>1</sub> )	liefert das Vorzeichen von a <sub>1</sub>
Min (a <sub>1</sub> , a <sub>2</sub> )	liefert den kleineren der beiden Argumente
Max (a <sub>1</sub> , a <sub>2</sub> )	liefert das größere der beiden Argumente
Pred (a <sub>1</sub> )	a <sub>1</sub> - 1.0
Succ (a <sub>1</sub> )	a <sub>1</sub> + 1.0
Adjust (a <sub>1</sub> , a <sub>2</sub> , a <sub>3</sub> )	reguliert a <sub>1</sub> in das Intervall [a <sub>2</sub> , a <sub>3</sub> ]
<b>trigonometrische Funktionen</b>	
AdjustPi (a <sub>1</sub> )	reguliert a <sub>1</sub> in ein [-π, π] Intervall
Adjust2Pi (a <sub>1</sub> )	reguliert a <sub>1</sub> in ein [-2π, 2π] Intervall
Adjust1 (a <sub>1</sub> )	reguliert a <sub>1</sub> in ein [-1.0, 1.0] Intervall
Arc (a <sub>1</sub> )	Arc (a <sub>1</sub> ) = a <sub>1</sub> * π / 180°
Sin (a <sub>1</sub> )	Sinus von a <sub>1</sub> , a <sub>1</sub> ist im Bogenmaß angegeben
Cos (a <sub>1</sub> )	Cosinus von a <sub>1</sub> , a <sub>1</sub> ist im Bogenmaß angegeben
Tan (a <sub>1</sub> )	Tangenz von a <sub>1</sub> , a <sub>1</sub> ist im Bogenmaß angegeben
CoTan (a <sub>1</sub> )	Cotangenz von a <sub>1</sub> , a <sub>1</sub> ist im Bogenmaß angegeben
Sec (a <sub>1</sub> )	Sekante von a <sub>1</sub> , a <sub>1</sub> ist im Bogenmaß angegeben
CoSec (a <sub>1</sub> )	Cosekante von a <sub>1</sub> , a <sub>1</sub> ist im Bogenmaß angegeben
ExSec (a <sub>1</sub> )	Exsekante von a <sub>1</sub> , a <sub>1</sub> ist im Bogenmaß angegeben
Vers (a <sub>1</sub> )	Versine von a <sub>1</sub> , a <sub>1</sub> ist im Bogenmaß angegeben
CoVers (a <sub>1</sub> )	Coversine von a <sub>1</sub> , a <sub>1</sub> ist im Bogenmaß angegeben
Hav (a <sub>1</sub> )	Haversine von a <sub>1</sub> , a <sub>1</sub> ist im Bogenmaß angegeben
<b>Umkehrfunktionen der trigonometrischen Funktionen</b>	
ArcSin (a <sub>1</sub> )	Arcussinus von a <sub>1</sub> , a <sub>1</sub> ist im Intervall [-1.0, 1.0] angegeben
ArcCos (a <sub>1</sub> )	Arcuscosinus von a <sub>1</sub> , a <sub>1</sub> ist im Intervall [-1.0, 1.0] angegeben
ArcTan (a <sub>1</sub> )	Arcustangenz von a <sub>1</sub>
ArcCoTan (a <sub>1</sub> )	Arcuscotangenz von a <sub>1</sub>
ArcSec (a <sub>1</sub> )	Arcussekante von a <sub>1</sub> , a <sub>1</sub> liegt außerhalb des Intervalls (-1.0, 1.0)
ArcCoSec (a <sub>1</sub> )	Arcuscosekante von a <sub>1</sub> , a <sub>1</sub> liegt außerhalb des Intervalls (-1.0, 1.0)
ArcExSec (a <sub>1</sub> )	Arcusexsekante von a <sub>1</sub> , a <sub>1</sub> liegt außerhalb des Intervalls (-2.0, 0.0)
ArcVers (a <sub>1</sub> )	Arcusversine von a <sub>1</sub> , a <sub>1</sub> liegt im Intervall [0.0, 2.0]
ArcCoVers (a <sub>1</sub> )	Arcuscoversine von a <sub>1</sub> , a <sub>1</sub> liegt im Intervall [0.0, 2.0]
ArcHav (a <sub>1</sub> )	Arcushaversine von a <sub>1</sub> , a <sub>1</sub> im Intervall [0.0, 1.0]
ArcSin2 (a <sub>1</sub> )	Die zweite Möglichkeit des Arcussinus von a <sub>1</sub> , a <sub>1</sub> liegt im

	Intervall [-1.0, 1.0]
ArcCos2 ( $a_1$ )	Die zweite Möglichkeit des Arcuscossinus von $a_1$ , $a_1$ liegt im Intervall [-1.0, 1.0]
ArcSec2 ( $a_1$ )	Die zweite Möglichkeit der Arcussekante von $a_1$ , $a_1$ liegt außerhalb des Intervalls (-1.0, 1.0)
ArcCoSec2 ( $a_1$ )	Die zweite Möglichkeit der Arcuscosekante von $a_1$ , $a_1$ liegt außerhalb des Intervalls (-1.0, 1.0)
ArcExSec2 ( $a_1$ )	Die zweite Möglichkeit der Arcussexsekante von $a_1$ , $a_1$ liegt außerhalb des Intervalls (-2.0, 0.0)
ArcVers2 ( $a_1$ )	Die zweite Möglichkeit der Arcusversine von $a_1$ , $a_1$ liegt im Intervall [0.0, 2.0]
ArcCoVers2 ( $a_1$ )	Die zweite Möglichkeit der Arcuscoversine von $a_1$ , $a_1$ liegt im Intervall [0.0, 2.0]
ArcHav2 ( $a_1$ )	Die zweite Möglichkeit der Arcushaversine von $a_1$ , $a_1$ liegt im Intervall [0.0, 1.0]
<b>hyperbolische Funktionen</b>	
Gd ( $a_1$ )	Die Gudermannfunktion von $a_1$
Sinh ( $a_1$ )	Hyperbolischer Sinus von $a_1$
Cosh ( $a_1$ )	Hyperbolischer Cosinus von $a_1$
Tanh ( $a_1$ )	Hyperbolischer Tangenz von $a_1$
CoTanh ( $a_1$ )	Hyperbolischer Cotangenz von $a_1$
Sech ( $a_1$ )	Hyperbolische Sekante von $a_1$
CoSech ( $a_1$ )	Hyperbolische Cosekante von $a_1$
<b>Umkehrfunktionen der hyperbolischen Funktionen</b>	
ArcGd ( $a_1$ )	Die Umkehrfunktion der Gudermannfunktion
ArcSinh ( $a_1$ )	Hyperbolischer Arcussinus von $a_1$
ArcCosh ( $a_1$ )	Hyperbolischer Arcuscossinus von $a_1$
ArcTanh ( $a_1$ )	Hyperbolischer Arcustangenz von $a_1$
ArcCoTanh ( $a_1$ )	Hyperbolischer Arcuscotangenz von $a_1$
ArcSech ( $a_1$ )	Hyperbolische Arcussekante von $a_1$
ArcCoSech ( $a_1$ )	Hyperbolische Arcuscosekante von $a_1$
ArcCosh2 ( $a_1$ )	Die zweite Möglichkeit des hyperbolischen Arcuscossinus von $a_1$
ArcSech2 ( $a_1$ )	Die zweite Möglichkeit der hyperbolischen Arcussekante von $a_1$
<b>kombinatorische Funktionen</b>	
Fakultaet ( $a_1$ )	Die Fakultät von $a_1$ ( $a_1!$ )
ArcFakultaet ( $a_1$ )	Die Umkehrfunktion der Fakultät von $a_1$
ArcFakultaet2 ( $a_1$ )	Die zweite Möglichkeit der Umkehrfunktion der Fakultät von $a_1$ (durch die „erweiterten mathematischen Gesetzmäßigkeiten“ © können sich unter Umständen zwei Möglichkeiten für die Arcusfakultät ergeben)
Binominal ( $a_1, a_2$ )	Der Binominalkoeffizient von $a_1$ über $a_2$
Permutation ( $a_1, a_2$ )	Anzahl der Variationen $a_2$ . Klasse von $a_1$ verschiedenen Elementen ohne Wiederholungen.
ZMZ ( $a_1, a_2, a_3$ )	Wahrscheinlichkeit beim Ziehen mit Zurücklegen: $a_2$ Versuche mit $a_3$ Erfolgen bei einer Erfolgswahrscheinlichkeit von $a_1$ pro Versuch
ZOZ ( $a_1, a_2, a_3, a_4$ )	Wahrscheinlichkeit beim ziehen ohne Zurücklegen: aus insgesamt $a_1$ Elementen, die $a_2$ gewünschte Elemente enthalten, werden $a_3$ Elemente gezogen. $a_4$ davon sind gewünschte Elemente.
NormalVerteilung ( $a_1$ )	Gaussche Verteilung von $a_1$ mit $\mu = 0.0$ und $\sigma^2 = 0.0$
GGT ( $a_1, a_2$ )	Der größte gemeinsame Teiler von $a_1$ und $a_2$
KGV ( $a_1, a_2$ )	Das kleinste gemeinsame Vielfache von $a_1$ und $a_2$
Divisors ( $a_1$ )	Anzahl der Divisoren von $a_1$
<b>orthogonale Funktionen</b>	
Legendre ( $a_1, a_2$ )	Legendrefunktion von $a_1$ ( $=n$ ) und $a_2$ ( $=x$ )
Tschebysheff1 ( $a_1, a_2$ )	Die erste Tschebyshefffunktion von $a_1$ ( $=n$ ) und $a_2$ ( $=x$ )

Tschebysheff2 ( $a_1, a_2$ )	Die zweite Tschebyshefffunktion von $a_1 (=n)$ und $a_2 (=x)$
Jacobi ( $a_1, a_2, a_3, a_4$ )	Die Jacobifunktion von $a_1 (=n)$ , $a_2 (=α)$ , $a_3 (=β)$ und $a_4 (=x)$
Laguerre ( $a_1, a_2, a_3$ )	Die Laguerrefunktion von $a_1 (=n)$ , $a_2 (=α)$ und $a_3 (=x)$
Hermite ( $a_1, a_2$ )	Die Hermitefunktion von $a_1 (=n)$ und $a_2 (=x)$
<b>TSF Fuzzylogikfunktionen</b>	
LFormat ( $a_1$ )	Reguliert $a_1$ in das Intervall [Negativ, Positiv]
LNot ( $a_1, a_2$ )	$\neg a_1$
LAnd ( $a_1, a_2$ )	$a_1 \wedge a_2$
LOr ( $a_1, a_1$ )	$a_1 \vee a_2$
LXOr ( $a_1, a_2$ )	$a_1 \text{ xor } a_2$ (entweder $a_1$ oder $a_2$ )
LCompare ( $a_1, a_2$ )	$a_1 > a_2?$
LPrime ( $a_1$ )	Ist $a_1$ eine Primzahl?
LGerade ( $a_1$ )	Ist $a_1$ eine gerade Zahl?
Random	Zufallszahl im Intervall [0.0, 1.0]

[Meine Besondere Lernleistung](#)  
[Users Manual](#)

# Der Zahentyp R

Zustand	Erklärung
Number	Reelle Zahl zwischen $-1.15 * 10^{4932}$ (MinR) und $1.15 * 10^{4932}$ (MaxR).
Unendlich ( $\infty$ )	Steht für Unendlich ( $\infty$ ).
nUnendlich ( $-\infty$ )	Steht für $-\infty$ .
MinOverFlow	Steht für eine reelle Zahl, die kleiner als $-1.15 * 10^{4932}$ (MinR) ist, und somit vom Computer nicht mehr dargestellt werden kann.
MaxOverFlow	Steht für eine reelle Zahl, die größer als $1.15 * 10^{4932}$ (MaxR) ist, und somit vom Computer nicht mehr dargestellt werden kann.
OverFlow	Steht für eine Zahl, die entweder MinOverFlow oder MaxOverFlow entspricht. (z.B. MinOverFlow + MaxOverFlow)
notDefined ( $\emptyset$ )	Steht für eine leere Menge, also einen nicht lösbaren Ausdruck wie z.B. $1 / 0$ .

## Die in Formula implementierten Funktionen

Funktion: Normalize  
 Eingabe:  $A_1$   
 Rückgabe: normalisiertes Argument

$A_1 =$	Rückgabe
Number	$A_1 < \text{MinR}$ : MinOverFlow $A_1 > \text{MaxR}$ : MaxOverFlow Anderenfalls: $A_1$
Unendlich	Unendlich
nUnendlich	nUnendlich
MinOverFlow	MinOverFlow
MaxOverFlow	MaxOverFlow
OverFlow	OverFlow

Funktion: Abs  
 Eingabe:  $A_1$   
 Rückgabe: Betrag des Argumentes

$A_1 =$	Rückgabe
Number	$ A_1 $
Unendlich	Unendlich
nUnendlich	Unendlich
MinOverFlow	MaxOverFlow
MaxOverFlow	MaxOverFlow
OverFlow	MaxOverFlow

Funktion: Neg  
 Eingabe:  $A_1$   
 Rückgabe:  $-A_1$

$A_1 =$	Rückgabe
Number	$-A_1$
Unendlich	nUnendlich
nUnendlich	Unendlich
MinOverFlow	MaxOverFlow

MaxOverflow	MinOverflow
Overflow	Overflow

Funktion: Int  
 Eingabe:  $A_1$   
 Rückgabe: Ganzzahlteil des Argumentes

$A_1 =$	Rückgabe
Number	Ganzzahlteil von $A_1$
Unendlich	Fehler: UnendlichOp Unendlich
nUnendlich	Fehler: UnendlichOp NUnendlich
MinOverflow	Fehler: OverflowOp MinOverflow
MaxOverflow	Fehler: OverflowOp MaxOverflow
Overflow	Fehler: OverflowOp Overflow

Funktion: Frac  
 Eingabe:  $A_1$   
 Rückgabe: Nachkommastelle des Argumentes

$A_1 =$	Rückgabe
Number	Nachkommastelle von $A_1$
Unendlich	Fehler: UnendlichOp notDefined
NUnendlich	Fehler: UnendlichOp notDefined
MinOverflow	Fehler: OverflowOp notDefined
MaxOverflow	Fehler: OverflowOp notDefined
Overflow	Fehler: OverflowOp notDefined

Funktion: Round  
 Eingabe:  $A_1$   
 Rückgabe: gerundetes Argument

$A_1 =$	Rückgabe
Number	$A_1 < 0$ : $\text{Frac}(A_1) \leq -0.5 \rightarrow \text{Int}(A_1) - 1$ anderenfalls $\text{Int}(A_1)$ $A_1 > 0$ : $\text{Frac}(A_1) \geq 0.5 \rightarrow \text{Int}(A_1) + 1$ anderenfalls $\text{Int}(A_1)$
Unendlich	Fehler: UnendlichOp Unendlich
nUnendlich	Fehler: UnendlichOp nUnendlich
MinOverflow	Fehler: OverflowOp MinOverflow
MaxOverflow	Fehler: OverflowOp MaxOverflow
Overflow	Fehler: OverflowOp Overflow

Funktion: HighRound

Eingabe:  $A_1$   
 Rückgabe: aufgerundetes Argument

$A_1 =$	Rückgabe
Number	$A_1 < 0$ : $\text{Frac}(A_1) < 0 \rightarrow \text{Int}(A_1) - 1$ anderenfalls $\text{Int}(A_1)$ $A_1 > 0$ : $\text{Frac}(A_1) > 0 \rightarrow \text{Int}(A_1) + 1$ anderenfalls $\text{Int}(A_1)$
Unendlich	Fehler: UnendlichOp Unendlich
NUnendlich	Fehler: UnendlichOp nUnendlich
MinOverFlow	Fehler: OverFlowOp MinOverFlow
MaxOverFlow	Fehler: OverFlowOp MaxOverFlow
OverFlow	Fehler: OverFlowOp OverFlow

Funktion: Add  
 Eingabe:  $A_1, A_2$   
 Rückgabe:  $A_1 + A_2$

	Number	Unendlich	nUnendlich	Minoverflow	Maxoverflow	Overflow
Number	$A_1 + A_2$	Unendlich	nUnendlich	Minoverflow	Maxoverflow	Overflow
Unendlich	Unendlich	Unendlich	NotDefined Fehler: UnendlichOp	Unendlich	Unendlich	Unendlich
NUnendlich						



Chemnitz, Freitag der 12. Februar 1999

**Thomas Weise**  
Schüler des Dr.-Wilhelm-André-Gymnasiums  
Klasse 11/6  
Tutorgruppe Herr Schmidt

## Besondere Lernleistung

Meine Besondere Lernleistung ist das Projekt „TS Formula“, bei dem die Hauptaufgabe darin besteht, ein Softwareprogramm zur Berechnung einer einzelnen gesuchten Größe aus einer Menge gegebener Größen zu entwickeln. Dafür sollen die notwendigen Zwischenformeln gesucht und ausgerechnet werden.

*Hieraus folgen zwei Teilaufgaben:*

1. Das Suchen der passenden Formeln und
2. deren Berechnung.

*Außerdem leiten sich folgende Aufgaben ab:*

1. Wie können Aufgabenstellungen sinnvoll gelöst werden?  
Hierbei spielt der Kontext der Formeln eine entscheidende Rolle, es wäre zum Beispiel unsinnig, Gleichungen der gleichförmigen und der beschleunigten Bewegung zu vermischen.  
Daraus wiederum folgt die Fragestellung: Wie kann man den Kontext der Formeln darstellen?
2. Diese Sachverhalte müssen in entsprechende Module des Basisprogramms umgesetzt werden.
3. Das Basisprogramm soll eine ansprechende Bedienoberfläche. Erhalten.
4. Außerdem sollen sämtliche Berechnungen, die das Programm durchführt, numerisch stabil sein. Es ist also notwendig, spezielle Ausnahmefälle zu erforschen (z.B. Division durch 0).
5. Die letzten drei Aufgaben erfordern Kenntnisse der Windows-Programmierung, die zu erlernen sind.

Bei der folgenden Erklärung meiner Besonderen Lernleistung beziehe ich mich auf die vom sächsischen Kultusministerium herausgegebene Broschüre „Besondere Lernleistung“, welche mir von Herrn König am Freitag, dem 05. Februar 1999 ausgehändigt wurde.

In dieser Broschüre wird erklärt: „Die rechtliche Grundlage für diese Neuerung wird für sächsische Gymnasien mit der Novellierung der Oberstufen- und Abiturprüfungsverordnung (OAVO vom 15.01.96) des Freistaates Sachsen geschaffen.“ (Seite zwei, Zeilen 25 bis 27)

Diese Besondere „Lernleistung ist ein selbst gewählter, auch selbst verantwortlicher Beitrag zu Erhöhung der Studierfähigkeit“ (Seite zwei, Zeilen 29 und 30), da dieses Projekt meine eigene Idee ist und auch meine Kenntnisse der Informatik und Mathematik erweitert und vertieft, was mir bei einem eventuellen Informatik- oder Mathematikstudium von Nutzen sein würde.

„Mit der Erarbeitung einer Besonderen Lernleistung werden kooperative Fähigkeiten entwickelt“, (Seite zwei, Zeilen 35 und 36) da dieses Projekt meine erste Kooperation gleichzeitig mit der Schule und der Technischen Universität Chemnitz-Zwickau darstellt.

Das Projekt „TS Formula“ betrifft das Themenfeld „Forschendes Lernen“ (Seite drei, Zeile 8), da ich während dem Projekt neue Kenntnisse und Methoden auf dem Gebiet der Informatik (speziell der Windows-Programmierung) und der Mathematik erlernen werde. Der forschende Bestandteil meiner Besonderen Lernleistung wird eine neue Umsetzung mathematischer Funktionen in die Informatik ebenso wie ein neuer Algorithmus beim Lösen tiefgehend verstrickter Formeln.

Die „Bedingungen für die Themenwahl“ (Seite drei, Zeile 1) werden wie folgt erfüllt:

„Die Zuordnungsmöglichkeit zu Unterrichtsfächern - auch im weiteren Sinne - im Interesse der Bewertbarkeit“ (Seite drei, Zeilen 12 und 13) ist gegeben, da das Projekt auf den Unterrichtsfächern Informatik, Mathematik und teilweise auch Physik beruht und diese vertieft.

„Der persönliche Bezug im Interesse der Förderung von Eigeninitiative und Kreativität und der Förderung selbstgesteuerten Lernens unter Anleitung“ (Seite drei, Zeilen 14 und 15) wird durch mein Interesse an der Programmierung (persönlicher Bezug) und durch die selbst entwickelte Idee hinter dem Projekt (Kreativität) sowie dadurch, dass die Betreuung hauptsächlich den theoretischen Teil (Mathematik, Informatik) meines Projektes betrifft, während ich den praktischen Teil (Windows-Programmieren) autodidaktisch erarbeiten werde (selbstgesteuertes Lernen unter Anleitung), erfüllt.

„Im Idealfall finden die Gymnasiasten ihre Themen selbst. Sie suchen sich geeignete Betreuer und Kooperationspartner und treffen selbständig die erforderlichen Absprachen.“ (Seite drei, Zeilen 17 und 18): dies trifft auf mein Projekt zu.

„Vor allem aus dem Unterricht - aus Grund- und Leistungskursen, fachübergreifenden Wahlgrundkursen, Projekten oder Praktika - können Themen abgeleitet werden.“ (Seite drei, Zeilen 19 und 20) wird erfüllt, da das Projekt den Leistungskurs Mathematik sowie die Grundkurse Informatik und Physik betrifft.

Das „zu erwarten ist, dass die Themen für die Besondere Lernleistung zunächst als Arbeitsthemen vorgelegt werden, die ihre Präzisierung durch den Arbeitsprozeß erfahren“ (Seite drei, Zeilen 28 und 29) ist die logische Konsequenz aus dem Prozeß der Entwicklung des Softwareprogramms „TS Formula“ im Rahmen der Besonderen Lernleistung.

Da der Informatiklehrer Herr König meine innerschulische und außerdem sekundäre Betreuung übernimmt, und mein Projekt die Informatik betrifft, ist auch eine Betreuung „durch einen geeigneten Fachlehrer“ (Seite vier, Zeile 2) gegeben.

„Bisherige Erfahrungen in Sachsen mit vergleichbaren, zusätzlich erbrachten, außergewöhnlichen Lernleistungen haben gezeigt, dass auch eine Begleitung durch externe Kooperationspartner sachdienlich ist. Die Zusammenarbeit mit mehreren Partnern ist möglich. Durch diese Begleitung wird zum einen der Realitätsbezug der Arbeit gestärkt, zum anderen erfahren die Schüler besonderes Interesse an ihrer Arbeit. Dadurch wird eine wichtige Motivationsgrundlage für weiteres eigenständiges Arbeiten gelegt.“ (Seite vier, Zeilen 5 bis 10). Mein externer Kooperationspartner ist Herr Dr. Gerd Kunert von der Technischen Universität Chemnitz-Zwickau. Er wird ebenfalls als primärer Betreuer tätig sein.

„Der geforderte ‚Umfang. Eines mindestens zweisemestrigen Kurses‘ (das bedeutet einen Arbeitsaufwand von mindestens 60 Stunden)“ (Seite vier, Zeilen 13 bis 15) wird bei dem Projekt „TS Formula“ durch „selbständige oder betreute Arbeit im schulischen oder außerschulischen Bereich“ (Seite vier, Zeile 20) erfüllt.

Da ich die Entscheidung, eine Besondere Lernleistung abzuliefern, bereits in den Sommerferien nach der zehnten Klassenstufe getroffen habe, und nun diese Entscheidung im zweiten Halbjahr der Klassenstufe elf noch einmal schriftlich bekräftige, ist auch „Die persönliche Entscheidung, eine Besondere Lernleistung erarbeiten zu wollen, treffen die Gymnasiasten in der Regel im Rahmen der Jahrgangsstufe 11.“ (Seite vier, Zeilen 21 und 22) erfüllt.

„Für die Erarbeitung einer Besonderen Lernleistung können sich die Schüler zwei Wochenstunden auf die zu erbringende Gesamtwochenstundenzahl anrechnen lassen. Entscheiden sich die Schüler für diese Anrechnung, wird die Arbeit an der Besonderen Lernleistung belegpflichtig, und es muss ein Ergebnis vorgelegt werden, das mit mindestens einem Punkt der einfachen Wertung bewertet wird - auch dann, wenn die Besondere Lernleistung nicht eingebracht werden soll“.

Spätestens zum Termin der Festlegung der Abiturprüfungsfächer aus dem Grundkursbereich entscheiden die Schüler verbindlich und auch bei Gruppenarbeiten individuell über die Einbringung ihrer Besonderen Lernleistung. Generell ist dabei zu beachten, dass die Bestandteile der Besonderen Lernleistung noch nicht anderweitig im Rahmen der Schule eingebracht worden sein dürfen.

Spätester Abgabetermin für die schriftliche Arbeit - gegebenenfalls mit fachpraktischer Komponente - der Besonderen Lernleistung ist der Termin des Halbjahreszeugnisses in Jahrgangsstufe 12. Das Kolloquium findet in der Regel im Rahmen der mündlichen Abiturprüfungen statt.

Die Erarbeitung einer Besonderen Lernleistung wird auf dem Abiturzeugnis auch dann zertifiziert, wenn der Schüler sie nicht in die Gesamtqualifikation einbringt.“  
(Seite vier, Zeilen 22 bis 39).

Hiermit bestätige ich, dass ich mein Projekt „TS Formula“ im Rahmen der Besondere Lernleistung gemäß dem obigen Zitat aus der Broschüre „Besondere Lernleistung“, vom sächsischen Kultusministerium herausgegeben, durchführen werde.

Meine besondere Lernleistung resultiert aus „der Aufarbeitung umfassender, auch fachübergreifender Projekte oder Praktika“ (Seite fünf, Zeile 6), nämlich aus dem Projekt „TS Formula“, welches die Fächer Mathematik, Informatik und Physik verknüpft.

„Die Erarbeitung von fachpraktischen Beiträge[n]“ (Seite fünf, Zeile 7) ist nicht nur „denkbar“, sondern auch Hauptbestandteil des Projekts.

„Bedingungen für die Anerkennung einer Arbeit als Besondere Lernleistung sind gezielte Aufarbeitung und systematische Reflexion von Arbeitsgegenstand, Arbeitsverlauf und Arbeitsergebnis. Diese Forderung gilt ausnahmslos für alle Themen.“ (Seite fünf, Zeilen 9 bis 11). Es gibt nur eine Möglichkeit, ein Programm zu schreiben: den Arbeitsgegenstand systematisch zu überdenken und dann die Probleme Schritt für Schritt zu lösen. Da ich dabei Protokoll führe, werde ich auch diese Bedingungen erfüllen.

## **Dokumentation**

Die so entstehende schriftliche Dokumentation, welche den folgenden Regeln entsprechen wird,

„Die Dokumentation enthält z.B.

- in der Einleitung: die Erläuterung und Abgrenzung des (möglichst originellen) Themas, die Begründung seiner Relevanz;
- im Hauptteil: den Nachweis der Verwendung angemessener Methoden, das geeignete Fixieren und die übersichtliche Darstellung der Ergebnisse sowie ggf. deren kritische Diskussion, eine kritische Methodenreflektion;
- im Schlußteil: die Darstellung möglicher Konsequenzen, Querverbindungen, Anwendungen und Auswirkungen.

Ein normgerechtes Quellenverzeichnis muss und ein möglicher Anhang sowie eine Kurzfassung sollten die Arbeit abschließen.“ (Seite fünf, Zeilen 17 bis 24),

wird dem Kolloquium vorgestellt werden.

Die Dokumentation wird auch den Bedingungen von Zitat Seite sechs, Zeilen 1 bis 12:

„Bewertungsgrundlage für die schriftliche Dokumentation ist der Nachweis der Beherrschung von Methoden, die auf wissenschaftliche Arbeitsweise vorbereiten. Dazu gehören:

- Wert und Umfang der Argumente;
- Konzentration auf das Wesentliche, Komprimiertheit; Präzision und logische Nachvollziehbarkeit der Darstellung
- Benennen der Gültigkeitsbedingungen der Ergebnisse
- Reflexion der Methoden und Lösungen - insbesondere bei mehreren möglichen Varianten;
- Originalität, Kreativität, Selbständigkeit und Problemorientierung;
- Qualität und Umfang der Recherchen;
- exakte Dokumentation des Arbeitsprozesses;
- Nachweis der Arbeitskontakte und Kooperationspartner.“

Da meine Besondere Lernleistung extern betreut wird, erfolgt die Bewertung der Dokumentation gemäß Zitat von Seite sechs, Zeilen 16 bis 19:

„Wurde die Arbeit extern betreut erfolgt die Erstkorrektur durch den externen Kooperationspartner. Aus rechtlichen und organisatorischen Gründen ist die Mitwirkung des betreuenden Lehrers bei der Beurteilung und Ermittlung der Leistungen unverzichtbar. Er übernimmt dann die Funktion des Zweitkorrektors.“

## **Kolloquium**

Abschließend werde ich meine Arbeit einem öffentlichen Kolloquium gemäß „Das abschließende, öffentliche Kolloquium dient der Präsentation des Arbeitsergebnisses. Die Schüler bzw. Schülergruppen weisen sich als Autoren der Arbeit mit fundierten Kenntnissen zu Zielen, Methoden, inhaltlichen Details und Ergebnissen aus.“ (Seite fünf, Zeilen 25 bis 27) vorgestellt werden.

Das Kolloquium wird von einer Prüfungskommission des Gymnasiums bewertet, zu der auch der Betreuer zählt. Da die Lernleistung extern betreut wird, stellt die Schule einen nicht stimmberechtigten Schriftführer.

„Bewertungsgrundlagen des Kolloquiums sind:

- Umfang des Wissens und Könnens;
- Argumentationssicherheit;
- Konzentration, Logik, Verständlichkeit der Ausführungen; Reaktionsfähigkeit, Engagement, Rhetorik;
- Sicherheit und Schauwert der Präsentation, wie z.B. fachpraktischer Vorführungen.“

(Seite sechs, Zeilen 27 bis 32)

„Sofern die Besondere Lernleistung eine fachpraktische Komponente enthält, gilt die Gewichtung: fachpraktische Komponente zu schriftliche Arbeit zu Kolloquium wie 1:1:1.“ (Seite sechs, Zeilen 40 und 41).

Die fachpraktische Komponente ist in Form einer, „durch zuständige Fachlehrer in ursächlichem Zusammenhang mit der Besonderen Lernleistung bewertete Arbeitsergebnisse aus Projekten, Praktika oder künstlerischer Tätigkeit“ (Seite sechs, Zeilen 43 und 44), nämlich das fertige Programm „TS Formula“.

Hiermit bestätige ich, dass ich das Projekt „TS Formula“ im Rahmen der Besonderen Lernleistung zu den oben genannten Bedingungen durchführen werde.

**Thomas Weise**

Schüler des Dr.-Wilhelm-André-Gymnasiums  
Klasse 11/6  
Tutorgruppe Herr Schmidt

Hiermit bestätige ich, dass ich als externer (außerschulischer) und gleichzeitig primärer Betreuer des Schülers Thomas Weise vom Dr. Wilhelm André Gymnasium bei seiner Besonderen Lernleistung, dem Projekt „TS Formula“, fungieren werde.

**Diplom-Mathematiker Dr. Gerd Kunert**

Technische Universität Chemnitz-Zwickau

Hiermit bestätige ich, dass ich als schulinterner und gleichzeitig sekundärer Betreuer des Schülers Thomas Weise vom Dr. Wilhelm André Gymnasium bei seiner Besonderen Lernleistung, dem Projekt „TS Formula“, fungieren werde.

**Raik König**

Informatiklehrer am Dr.-Wilhelm-André-Gymnasium

Hiermit genehmige ich die Besondere Lernleistung „TS Formula“ des Schüler Thomas Weise.

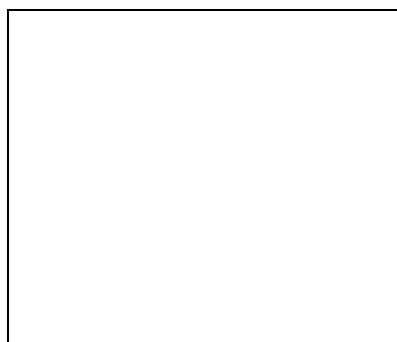
**Gudrun Löschner**

Schulleiterin des Dr.-Wilhelm-André-Gymnasium

**Ute Borowiki**

Zuständige für Sachen die Besondere Lernleistung  
betreffend am Dr.-Wilhelm-André-Gymnasium

**Schulstempel**





# Grobe Planung

Einen Zeitplan zu erstellen ist für ein Projekt, in dem es hauptsächlich um Programmierung geht, unmöglich, da man nie weiß, an welcher Stelle Probleme auftreten werden. Man kann nur die grobe Dauer insgesamt voraussagen.

Es ist jedoch möglich, eine Liste der Teilaufgaben sowie deren Reihenfolge zu generieren.

1. Erlernen der Programmiersprache Borland Delphi im notwendigen Umfang
2. Erstellen eines Objekts, welches eine Form darstellt
3. Erstellen eines Objekts, welches eine Datenbasis repräsentiert, in die Formelobjekte gespeichert und gelesen werden können
4. Erstellen einer Eingabemaske für Formeln in die Datenbasis
5. Erstellen eines Modules zur Umwandlung von Formelobjekten in PASCAL-Code und deren Speicherung, so das der eingebauten Delphi-Compiler lauffähige DLL's erstellen kann
6. Erstellen einer Maske zur Eingabe für die Lösung von Formeln
7. Erstellen eines Lösungsalgorithmus und Erweitern des Moduls zur Umwandlung von Formelobjekten zum Laden von DLL's
8. Implementierung der mathematischen Grundfunktionen
9. Verbinden der beiden Eingabemasken zu einer formschönen und einfachen Oberfläche
10. Den theoretischen Teil der Besonderen Lernleistung anfertigen

Optional:

11. Versetzen des Programms mit einer Hilfe-Funktion
12. Erstellen einer Animation des Programms für das Kolloquium
13. Ausgabe des Programms an Testpersonen

Erledigt

In Arbeit

Vorgesehen

Änderungen Vorbehalt, Thomas Weise, 27.06.1999



Chemnitz, Sonntag der 27. Juni 1999

**Thomas Weise**  
Schüler des Dr.-Wilhelm-André-Gymnasiums  
Klasse 11/6  
Tutorgruppe Herr Schmidt

## Besondere Lernleistung

Meine Besondere Lernleistung ist das Projekt „TS Formula“, bei dem die Hauptaufgabe darin besteht, ein Softwareprogramm zur Berechnung einer einzelnen gesuchten Größe aus einer Menge gegebener Größen zu entwickeln. Dafür sollen die notwendigen Zwischenformeln gesucht und ausgerechnet werden.

*Hieraus folgen zwei Teilaufgaben:*

1. Das Suchen der passenden Formeln und
2. deren Berechnung.

*Außerdem leiten sich folgende Aufgaben ab:*

1. Wie können Aufgabenstellungen sinnvoll gelöst werden?  
Hierbei spielt der Kontext der Formeln eine entscheidende Rolle, es wäre zum Beispiel unsinnig, Gleichungen der gleichförmigen und der beschleunigten Bewegung zu vermischen.  
Daraus wiederum folgt die Fragestellung: Wie kann man den Kontext der Formeln darstellen?
2. Diese Sachverhalte müssen in entsprechende Module des Basisprogramms umgesetzt werden.
3. Das Basisprogramm soll eine ansprechende Bedienoberfläche. Erhalten.
4. Außerdem sollen sämtliche Berechnungen, die das Programm durchführt, numerisch stabil sein. Es ist also notwendig, spezielle Ausnahmefälle zu erforschen (z.B. Division durch 0).
5. Die letzten drei Aufgaben erfordern Kenntnisse der Windows-Programmierung, die zu erlernen sind.

Bei der folgenden Erklärung meiner Besonderen Lernleistung beziehe ich mich auf die vom sächsischen Kultusministerium herausgegebene Broschüre „Besondere Lernleistung“, welche mir von Herrn König am Freitag, dem 05. Februar 1999 ausgehändigt wurde.

In dieser Broschüre wird erklärt: „Die rechtliche Grundlage für diese Neuerung wird für sächsische Gymnasien mit der Novellierung der Oberstufen- und Abiturprüfungsverordnung (OAVO vom 15.01.96) des Freistaates Sachsen geschaffen.“ (Seite zwei, Zeilen 25 bis 27)

Diese Besondere „Lernleistung ist ein selbst gewählter, auch selbst verantwortlicher Beitrag zu Erhöhung der Studierfähigkeit“ (Seite zwei, Zeilen 29 und 30), da dieses Projekt meine eigene Idee ist und auch meine Kenntnisse der Informatik und Mathematik erweitert und vertieft, was mir bei einem eventuellen Informatik- oder Mathematikstudium von Nutzen sein würde.

„Mit der Erarbeitung einer Besonderen Lernleistung werden kooperative Fähigkeiten entwickelt“, (Seite zwei, Zeilen 35 und 36) da dieses Projekt meine erste Kooperation gleichzeitig mit der Schule und der Technischen Universität Chemnitz-Zwickau darstellt.

Das Projekt „TS Formula“ betrifft das Themenfeld „Forschendes Lernen“ (Seite drei, Zeile 8), da ich während dem Projekt neue Kenntnisse und Methoden auf dem Gebiet der Informatik (speziell der Windows-Programmierung) und der Mathematik erlernen werde. Der forschende Bestandteil meiner Besonderen Lernleistung wird eine neue Umsetzung mathematischer Funktionen in die Informatik ebenso wie ein neuer Algorithmus beim Lösen tiefgehend verstrickter Formeln.

Die „Bedingungen für die Themenwahl“ (Seite drei, Zeile 1) werden wie folgt erfüllt:

„Die Zuordnungsmöglichkeit zu Unterrichtsfächern - auch im weiteren Sinne - im Interesse der Bewertbarkeit“ (Seite drei, Zeilen 12 und 13) ist gegeben, da das Projekt auf den Unterrichtsfächern Informatik, Mathematik und teilweise auch Physik beruht und diese vertieft.

„Der persönliche Bezug im Interesse der Förderung von Eigeninitiative und Kreativität und der Förderung selbstgesteuerten Lernens unter Anleitung“ (Seite drei, Zeilen 14 und 15) wird durch mein Interesse an der Programmierung (persönlicher Bezug) und durch die selbst entwickelte Idee hinter dem Projekt (Kreativität) sowie dadurch, dass die Betreuung hauptsächlich den theoretischen Teil (Mathematik, Informatik) meines Projektes betrifft, während ich den praktischen Teil (Windows-Programmieren) autodidaktisch erarbeiten werde (selbstgesteuertes Lernen unter Anleitung), erfüllt.

„Im Idealfall finden die Gymnasiasten ihre Themen selbst. Sie suchen sich geeignete Betreuer und Kooperationspartner und treffen selbständig die erforderlichen Absprachen.“ (Seite drei, Zeilen 17 und 18): dies trifft auf mein Projekt zu.

„Vor allem aus dem Unterricht - aus Grund- und Leistungskursen, fachübergreifenden Wahlgrundkursen, Projekten oder Praktika - können Themen abgeleitet werden.“ (Seite drei, Zeilen 19 und 20) wird erfüllt, da das Projekt den Leistungskurs Mathematik sowie die Grundkurse Informatik und Physik betrifft.

Das „zu erwarten ist, dass die Themen für die Besondere Lernleistung zunächst als Arbeitsthemen vorgelegt werden, die ihre Präzisierung durch den Arbeitsprozeß erfahren“ (Seite drei, Zeilen 28 und 29) ist die logische Konsequenz aus dem Prozeß der Entwicklung des Softwareprogramms „TS Formula“ im Rahmen der Besonderen Lernleistung.

Da der Informatiklehrer Herr König meine innerschulische und außerdem sekundäre Betreuung übernimmt, und mein Projekt die Informatik betrifft, ist auch eine Betreuung „durch einen geeigneten Fachlehrer“ (Seite vier, Zeile 2) gegeben.

„Bisherige Erfahrungen in Sachsen mit vergleichbaren, zusätzlich erbrachten, außergewöhnlichen Lernleistungen haben gezeigt, dass auch eine Begleitung durch externe Kooperationspartner sachdienlich ist. Die Zusammenarbeit mit mehreren Partnern ist möglich. Durch diese Begleitung wird zum einen der Realitätsbezug der Arbeit gestärkt, zum anderen erfahren die Schüler besonderes Interesse an ihrer Arbeit. Dadurch wird eine wichtige Motivationsgrundlage für weiteres eigenständiges Arbeiten gelegt.“ (Seite vier, Zeilen 5 bis 10). Mein externer Kooperationspartner ist Herr Dr. Gerd Kunert von der Technischen Universität Chemnitz-Zwickau. Er wird ebenfalls als primärer Betreuer tätig sein.

„Der geforderte ‚Umfang. Eines mindestens zweisemestrigen Kurses‘ (das bedeutet einen Arbeitsaufwand von mindestens 60 Stunden)“ (Seite vier, Zeilen 13 bis 15) wird bei dem Projekt „TS Formula“ durch „selbständige oder betreute Arbeit im schulischen oder außerschulischen Bereich“ (Seite vier, Zeile 20) erfüllt.

Da ich die Entscheidung, eine Besondere Lernleistung abzuliefern, bereits in den Sommerferien nach der zehnten Klassenstufe getroffen habe, und nun diese Entscheidung im zweiten Halbjahr der Klassenstufe elf noch einmal schriftlich bekräftige, ist auch „Die persönliche Entscheidung, eine Besondere Lernleistung erarbeiten zu wollen, treffen die Gymnasiasten in der Regel im Rahmen der Jahrgangsstufe 11.“ (Seite vier, Zeilen 21 und 22) erfüllt.

„Für die Erarbeitung einer Besonderen Lernleistung können sich die Schüler zwei Wochenstunden auf die zu erbringende Gesamtwochenstundenzahl anrechnen lassen. Entscheiden sich die Schüler für diese Anrechnung, wird die Arbeit an der Besonderen Lernleistung belegpflichtig, und es muss ein Ergebnis vorgelegt werden, das mit mindestens einem Punkt der einfachen Wertung bewertet wird - auch dann, wenn die Besondere Lernleistung nicht eingebracht werden soll“.

Spätestens zum Termin der Festlegung der Abiturprüfungsfächer aus dem Grundkursbereich entscheiden die Schüler verbindlich und auch bei Gruppenarbeiten individuell über die Einbringung ihrer Besonderen Lernleistung. Generell ist dabei zu beachten, dass die Bestandteile der Besonderen Lernleistung noch nicht anderweitig im Rahmen der Schule eingebracht worden sein dürfen.

Spätester Abgabetermin für die schriftliche Arbeit - gegebenenfalls mit fachpraktischer Komponente - der Besonderen Lernleistung ist der Termin des Halbjahreszeugnisses in Jahrgangsstufe 12. Das Kolloquium findet in der Regel im Rahmen der mündlichen Abiturprüfungen statt.

Die Erarbeitung einer Besonderen Lernleistung wird auf dem Abiturzeugnis auch dann zertifiziert, wenn der Schüler sie nicht in die Gesamtqualifikation einbringt.“  
(Seite vier, Zeilen 22 bis 39).

Hiermit bestätige ich, dass ich mein Projekt „TS Formula“ im Rahmen der Besondere Lernleistung gemäß dem obigen Zitat aus der Broschüre „Besondere Lernleistung“, vom sächsischen Kultusministerium herausgegeben, durchführen werde.

Meine besondere Lernleistung resultiert aus „der Aufarbeitung umfassender, auch fachübergreifender Projekte oder Praktika“ (Seite fünf, Zeile 6), nämlich aus dem Projekt „TS Formula“, welches die Fächer Mathematik, Informatik und Physik verknüpft.

„Die Erarbeitung von fachpraktischen Beiträge[n]“ (Seite fünf, Zeile 7) ist nicht nur „denkbar“, sondern auch Hauptbestandteil des Projekts.

„Bedingungen für die Anerkennung einer Arbeit als Besondere Lernleistung sind gezielte Aufarbeitung und systematische Reflexion von Arbeitsgegenstand, Arbeitsverlauf und Arbeitsergebnis. Diese Forderung gilt ausnahmslos für alle Themen.“ (Seite fünf, Zeilen 9 bis 11). Es gibt nur eine Möglichkeit, ein Programm zu schreiben: den Arbeitsgegenstand systematisch zu überdenken und dann die Probleme Schritt für Schritt zu lösen. Da ich dabei Protokoll führe, werde ich auch diese Bedingungen erfüllen.

## **Dokumentation**

Die so entstehende schriftliche Dokumentation, welche den folgenden Regeln entsprechen wird,

„Die Dokumentation enthält z.B.

- in der Einleitung: die Erläuterung und Abgrenzung des (möglichst originellen) Themas, die Begründung seiner Relevanz;
- im Hauptteil: den Nachweis der Verwendung angemessener Methoden, das geeignete Fixieren und die übersichtliche Darstellung der Ergebnisse sowie ggf. deren kritische Diskussion, eine kritische Methodenreflektion;
- im Schlußteil: die Darstellung möglicher Konsequenzen, Querverbindungen, Anwendungen und Auswirkungen.

Ein normgerechtes Quellenverzeichnis muss und ein möglicher Anhang sowie eine Kurzfassung sollten die Arbeit abschließen.“ (Seite fünf, Zeilen 17 bis 24),

wird dem Kolloquium vorgestellt werden.

Die Dokumentation wird auch den Bedingungen von Zitat Seite sechs, Zeilen 1 bis 12:

„Bewertungsgrundlage für die schriftliche Dokumentation ist der Nachweis der Beherrschung von Methoden, die auf wissenschaftliche Arbeitsweise vorbereiten. Dazu gehören:

- Wert und Umfang der Argumente;
- Konzentration auf das Wesentliche, Komprimiertheit; Präzision und logische Nachvollziehbarkeit der Darstellung
- Benennen der Gültigkeitsbedingungen der Ergebnisse
- Reflexion der Methoden und Lösungen - insbesondere bei mehreren möglichen Varianten;
- Originalität, Kreativität, Selbständigkeit und Problemorientierung;
- Qualität und Umfang der Recherchen;
- exakte Dokumentation des Arbeitsprozesses;
- Nachweis der Arbeitskontakte und Kooperationspartner.“

Da meine Besondere Lernleistung extern betreut wird, erfolgt die Bewertung der Dokumentation gemäß Zitat von Seite sechs, Zeilen 16 bis 19:

„Wurde die Arbeit extern betreut erfolgt die Erstkorrektur durch den externen Kooperationspartner. Aus rechtlichen und organisatorischen Gründen ist die Mitwirkung des betreuenden Lehrers bei der Beurteilung und Ermittlung der Leistungen unverzichtbar. Er übernimmt dann die Funktion des Zweitkorrektors.“

## **Kolloquium**

Abschließend werde ich meine Arbeit einem öffentlichen Kolloquium gemäß „Das abschließende, öffentliche Kolloquium dient der Präsentation des Arbeitsergebnisses. Die Schüler bzw. Schülergruppen weisen sich als Autoren der Arbeit mit fundierten Kenntnissen zu Zielen, Methoden, inhaltlichen Details und Ergebnissen aus.“ (Seite fünf, Zeilen 25 bis 27) vorgestellt werden.

Das Kolloquium wird von einer Prüfungskommission des Gymnasiums bewertet, zu der auch der Betreuer zählt. Da die Lernleistung extern betreut wird, stellt die Schule einen nicht stimmberechtigten Schriftführer.

„Bewertungsgrundlagen des Kolloquiums sind:

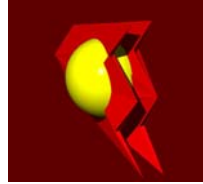
- Umfang des Wissens und Könnens;
- Argumentationssicherheit;
- Konzentration, Logik, Verständlichkeit der Ausführungen; Reaktionsfähigkeit, Engagement, Rhetorik;
- Sicherheit und Schauwert der Präsentation, wie z.B. fachpraktischer Vorführungen.“

(Seite sechs, Zeilen 27 bis 32)

„Sofern die Besondere Lernleistung eine fachpraktische Komponente enthält, gilt die Gewichtung: fachpraktische Komponente zu schriftliche Arbeit zu Kolloquium wie 1:1:1.“ (Seite sechs, Zeilen 40 und 41).

Die fachpraktische Komponente ist in Form einer, „durch zuständige Fachlehrer in ursächlichem Zusammenhang mit der Besonderen Lernleistung bewertete Arbeitsergebnisse aus Projekten, Praktika oder künstlerischer Tätigkeit“ (Seite sechs, Zeilen 43 und 44), nämlich das fertige Programm „TS Formula“.

Änderungen Vorbehalt, Thomas Weise, 27.06.1999



Thomas Weise  
Tschaikowskistraße 30  
09130 Chemnitz  
☎ (0371) 4015607  
☎ (0371) 4015607

## Meine besondere Lernleistung

### „TS Formula“

Meine besondere Lernleistung betrifft das Themenfeld „Forschendes Lernen“ und fällt fächertübergreifend in die Informatik, Mathematik und die Physik.

Die Leistung wird durch Verwendung der Programmiersprache Delphi 4.0 Standard (bzw. diese Alphaversion in Turbo Pascal 7.0) erbracht.

Die Ziele meiner besonderen Lernleistung sind:

- Das Erstellen eines umfassenden mathematischen Formelkataloges für den Bereich der reellen Zahlen
- Die Schaffung eines Programms, was benutzerfreundliche Ein- und Ausgabe, Bearbeitung sowie die Lösung von Formeln gestattet
- Die Erweiterung diese Programms um eine Benutzerfreundlichere Oberfläche
- Die Erstellung einer Datenbank, die möglichst viele Formeln aus dem Bereich der Physik enthält

Selbst gestellte Anforderungen sind:

- Die (möglichst) vollständige Kapselung aller verwendeter Objekte
- Möglichst schnelles Lösen sämtlicher Aufgaben
- Möglichst hohe Absturzsicherheit (den Windows-Faktor kann ich leider nicht besiegen)
- Möglichst wenige der von Delphi bzw. Pascal definierten Prozeduren und Funktionen verwenden
- Möglichst hohe Narrensicherheit
- Betriebssystemnahe Programmierung (mit DOS in Alphaversion realisiert, bei Windows in Arbeit)
- Logisches Lösen von Aufgaben, teilweise mit Hilfe von KI

### Was ist TS Formula?

TS Formula erklärt sich meiner Meinung praktisch von selbst bei der Benutzung, es dient dazu komplizierte physikalische und mathematische Probleme teilweise mit hinzunahme von KI im Rahmen der logischen Erkennung von Problemen und deren Lösung durch angewandte Logik.

Es stellt eine hilfreiche Stütze bei Berechnungen dar, in denen mehrere Ebenen tief auf verschiedene Funktionen zugegriffen werden soll (was gemeint ist wird bei der Benutzung klar).

### Wie funktioniert TS Formula?

TS Formula löst Probleme indem es auf sein gespeichertes Wissen zugreift. Dieses Wissen wird dem Programm durch den Benutzer zugeführt. Er gibt ihm Formeln ein (in der Alphaversion mit Hilfe des „DATAREAD“ Programms), die Formula dann in seiner Datenbasis („FORMELN.DBQ“) ablegt. Will der Benutzer nun etwas einen bestimmten Wert berechnen, so speist er die bekannten Werte ein und das Programm such dann in seinem „Wissen“ nach einer passenden Regel und wendet diese an, um das Problem zu lösen (in der Alphaversion macht das das „FORMEL“ Programm).

### **Ist das so einfach?**

Nein, ganz so einfach ist es nun auch wieder nicht: bei der Eingabe von Formeln (in der Alphaversion) muß man bestimmte Eigenheiten des Programms beachten:

1. Man kann nur auf vordefinierte Funktionen (in der „REEL.TPU“) zugreifen, einfache Addition durch das „+“ Zeichen ist nicht möglich, man muß schon die reele Funktion „Add (a<sub>1</sub>, a<sub>2</sub>)“ verwenden.
2. Die Parameter müssen stets angegeben werden, da sie das Programm nicht selbst herausfinden kann.
3. Berechnungen sind nur im Bereich der reellen Zahlen möglich.
4. Das Programm ist nicht in der Lage, Formeln selbst umzustellen, es ist daher günstig, sie immer nach allen Parametern umgestellt einzugeben.

Das klingt jetzt sehr kompliziert, ist es aber nicht, wie sie in der Anleitung erkennen werden. Und mit etwas Glück und (mehr) Arbeit kann man das in der Windowsversion auch beheben.

### **„erweiterte mathematischen Gesetzmäßigkeiten“?**

Die „erweiterten mathematischen Gesetzmäßigkeiten“ sind ein fester Bestandteil der KI von TS Formula, sie dienen dazu mathematische Grauzonen an die Praxis anzupassen.

Wurde bei einer Berechnung eine solche Anpassung vorgenommen, so wird dies natürlich kenntlich gemacht, und es bleibt dem Benutzer überlassen, ob er das Ergebnis werten will oder nicht.

Wie ist das zu verstehen? Am besten kann ich das an einem Beispiel erklären: Es sei die Formel  $R = \text{Divi}(U, I)$  mit den Parametern U und I gegeben (entspricht  $R = U/I$ ). Was passiert, wenn I gegen 0 geht? Mathematisch wäre die Aufgabe nicht lösbar, praktisch ist R aber unendlich groß. Und genau dieses Ergebnis würden sie auch mit TS Formula erhalten: „Das Ergebnis geht gegen Unendlich. Es trat eine Division durch Null auf!“. Nun kann der Benutzer entscheiden: „Das Problem war rein mathematischer Natur, also ist die Aufgabe nicht lösbar!“ oder „Ich brauche ein praktisches Ergebnis, also nützt mir diese Auskunft etwas!“. Ähnliches passiert auch bei  $\text{Tan}(\text{Divi}(\pi, 2))$  usw.

(und beim Tangenz tritt mein erstes schweres Windows-Problem auf:  $\text{Sin}(\pi) = -5.42101\text{E}-20!$ )

### **Wie soll es gewertet werden?**

Die aufgebrauchten Stunden (sicher schon über 300) sollen nicht von der Schulzeit abgerechnet werden, und ob ich es als besondere Lernleistung gelten lassen will, lasse ich erstmal auf ihr vorläufiges Urteil ankommen.

[Users Manual](#)

[Die implementierten Funktionen](#)

Liebe Schülerinnen und Schüler,

die Abiturienten des Jahrgangs 2000 sind die ersten, die die Möglichkeit haben, eine Besondere Lernleistung in den Abiturprüfungsbereich der Gesamtqualifikation einzubringen.

Die Besondere Lernleistung ist ein Angebot für kreative junge Leute. Mit der Erarbeitung und Einbringung einer Besonderen Lernleistung haben Sie es in der Hand, Prozeß und Ergebnis Ihrer allgemeinen Hochschulreife zu beeinflussen.

Lassen Sie sich beraten.

Die Entscheidung, sich dieser Aufgabe, allein oder mit anderen, zu stellen oder nicht, liegt bei Ihnen. Die Entscheidung über das Thema Ihrer Arbeit liegt bei Ihnen. Die Entscheidung über die Betreuung Ihrer Arbeit, das Einverständnis des von Ihnen anvisierten Partners vorausgesetzt, liegt bei Ihnen. Die Entscheidung, sich die Arbeit auf Ihr Regelstundenmaß anrechnen zu lassen, liegt bei Ihnen. Sie selbst verantworten auch die Einbringung des Ihnen noch nicht genau bekannten Ergebnisses in den Abiturprüfungsbereich.

Die Besondere Lernleistung ist also Chance und Herausforderung zugleich, eben eine besondere Art der Abiturvorbereitung.

Die folgenden Informationen wollen Sie mit den Bedingungen dieser neuen Möglichkeit vertraut machen.

Für die Arbeit an der Besonderen Lernleistung wünsche ich Ihnen jenes geistige Vergnügen, das aus Anstrengung erwächst, und viel Erfolg.

Dr. Matthias Rößler

# Besondere Lernleistung

## 1. Rahmenbedingungen

Die Vereinbarung der Kultusministerkonferenz vom Februar 1997 führt zur Besonderen Lernleistung aus:

„Im Rahmen der für die Abiturprüfung vorgesehenen Gesamtpunktzahl können die Länder vorsehen, daß Schülerinnen und Schüler wahlweise eine besondere Lernleistung, die im Rahmen bzw. Umfang eines mindestens zweisemestrigen Kurses erbracht wird, in die Abiturprüfung einbringen können. Besondere Lernleistungen können z.B. sein: Ein umfassender Beitrag aus einem von den Ländern geförderten Wettbewerb, eine Jahres- oder Seminararbeit, die Ergebnisse eines umfassenden, auch fachübergreifenden Projektes oder Praktikums in Bereichen, die schulischen Referenzfächern zugeordnet werden können. Die besondere Lernleistung ist schriftlich zu dokumentieren. Voraussetzung für die Einbringung ist, daß die besondere Lernleistung oder wesentliche Bestandteile noch nicht anderweitig im Rahmen der Schule angerechnet wurden. In einem Kolloquium stellt die Schülerin bzw. der Schüler die Ergebnisse der besonderen Lernleistung dar; erläutert sie und antwortet auf Fragen. Bei Arbeiten, an denen mehrere Schülerinnen und Schüler beteiligt waren, ist die Bewertung der individuellen Schülerleistung erforderlich.“

Weiter heißt es in dieser Vereinbarung:

„In der Abiturprüfung sind in diesem Fall in den vier Abiturprüfungsfächern maximal jeweils 60 Punkte erreichbar. Dabei sind die Leistungen in diesen Fächern im letzten Schulhalbjahr jeweils einfach, die in der Abiturprüfung erbrachten Leistungen jeweils dreifach zu werten. In der besonderen Lernleistung sind maximal 15 Punkte in vierfacher Wertung erreichbar.“

Die rechtliche Grundlage für diese Neuerung wird für sächsische Gymnasien mit der Novellierung der Oberstufen- und Abiturprüfungsverordnung (OAVO vom 15.01.96) des Freistaates Sachsen geschaffen.

## 2. Ziele

Die Erarbeitung einer Besonderen Lernleistung ist ein selbst gewählter, aber auch selbst verantworteter Beitrag zur Erhöhung der Studierfähigkeit.

Die Erarbeitung einer Besonderen Lernleistung wird größere Klarheit über eigenes Arbeitsverhalten und über Breite und Tiefe eigener Interessen bringen.

Die Erarbeitung einer Besonderen Lernleistung wird helfen, Studien- und Berufseingangsvoraussetzungen zu verbessern.

Mit der Erarbeitung einer Besonderen Lernleistung werden kooperative Fähigkeiten entwickelt.

## 3. Themen

### 3.1. Themenfelder

Ausgehend von O.g. Zielstellung erschließt sich ein breites Handlungsfeld für eigenverantwortetes Lernen.

Kooperation mit außerschulischen Partnern wie Hochschulen, Verbänden, Unternehmen, Kirchen, politischen und sozialen Einrichtungen wird nicht nur akzeptiert, sondern gefördert.

Themenfelder sind:

- forschendes Lernen,
- Künstlerische Tätigkeit,
- Demokratisch-soziales Handeln.

Bedingungen für die Themenwahl sind:

- die Zuordnungsmöglichkeit zu Unterrichtsfächern - auch im weiteren Sinne - im Interesse der Bewertbarkeit
- der persönliche Bezug im Interesse der Förderung von Eigeninitiative und Kreativität und der Förderung selbstgesteuerten Lernens unter Anleitung.

### 3.2. Themenfindung

Im Idealfall finden die Gymnasiasten ihre Themen selbst. Sie suchen sich geeignete Betreuer und Kooperationspartner und treffen selbständig die erforderlichen Absprachen.

Vor allem aus dem Unterricht - aus Grund- und Leistungskursen, fachübergreifenden Wahlgrundkursen, Projekten oder Praktika - können Themen abgeleitet werden.

Zur Themenfindung für Besondere Lernleistungen sind auch geeignet:

- Mitarbeit an Projekten von Hochschulen, Instituten, Unternehmen;
- Mitarbeit an künstlerischen Projekten;
- Mitarbeit an ökonomischen, ökologischen, sozialen und anderen gesellschaftlichen Projekten;
- Engagement in Vereinen und Verbänden;
- Exkursionen.

Zu erwarten ist, dass die Themen für die Besondere Lernleistung zunächst als Arbeitsthemen vorgelegt werden, die ihre Präzisierung durch den Arbeitsprozeß erfahren.

### 3.3. Themenangebote

Zunächst bieten die vom Freistaat geförderten Schüler-Wettbewerbe Themen für die Besondere Lernleistung. Weitere Möglichkeiten ergeben sich aus Angeboten der Hochschulen. Die Gymnasien mit vertiefter Ausbildung verpflichten sich, ihren Schülern geeignete Angebote zu unterbreiten.

#### **4. Betreuung**

Der Schüler wird durch einen geeigneten Fachlehrer betreut. Die Schule hat gegenüber Schülern, die sich für eine Besondere Lernleistung entscheiden, eine Beratungs- und Betreuungspflicht.

Bisherige Erfahrungen in Sachsen mit vergleichbaren, zusätzlich erbrachten, außergewöhnlichen Lernleistungen haben gezeigt, dass auch eine Begleitung durch externe Kooperationspartner sachdienlich ist. Die Zusammenarbeit mit mehreren Partnern ist möglich. Durch diese Begleitung wird zum einen der Realitätsbezug der Arbeit gestärkt, zum anderen erfahren die Schüler besonderes Interesse an ihrer Arbeit. Dadurch wird eine wichtige Motivationsgrundlage für weiteres eigenständiges Arbeiten gelegt.

#### **5. Belegung und Einbringung**

Der geforderte „Umfang eines mindestens zweisemestrigen Kurses (das bedeutet einen Arbeitsaufwand von mindestens 60 Stunden) kann auf verschiedene Weise realisiert und abgerechnet werden.

Denkbar sind:

- Fortführung eines in einem Pflicht- oder Wahlgrundkurs begonnenen Projektes im außerunterrichtlichen Bereich und in Projektwochen;
- Belegung eines Wahlgrundkurses oder einer Arbeitsgemeinschaft;
- selbständige oder betreute Arbeit im schulischen oder außerschulischen Bereich.

Die persönliche Entscheidung, eine Besondere Lernleistung erarbeiten zu wollen, treffen die Gymnasiasten in der Regel im Rahmen der Jahrgangsstufe 11. Für die Erarbeitung einer Besonderen Lernleistung können sich die Schüler zwei Wochenstunden auf die zu erbringende Gesamtwochenstundenzahl anrechnen lassen. Entscheiden sich die Schüler für diese Anrechnung, wird die Arbeit an der Besonderen Lernleistung belegpflichtig, und es muss ein Ergebnis vorgelegt werden, das mit mindestens einem Punkt der einfachen Wertung bewertet wird - auch dann, wenn die Besondere Lernleistung nicht eingebracht werden soll.

Spätestens zum Termin der Festlegung der Abiturprüfungsfächer aus dem Grundkursbereich entscheiden die Schüler verbindlich und auch bei Gruppenarbeiten individuell über die Einbringung ihrer Besonderen Lernleistung.

Generell ist dabei zu beachten, dass die Bestandteile der Besonderen Lernleistung noch nicht anderweitig im Rahmen der Schule eingebracht worden sein dürfen.

Spätester Abgabetermin für die schriftliche Arbeit - gegebenenfalls mit fachpraktischer Komponente - der Besonderen Lernleistung ist der Termin des Halbjahreszeugnisses in Jahrgangsstufe 12. Das Kolloquium findet in der Regel im Rahmen der mündlichen Abiturprüfungen statt.

Die Erarbeitung einer Besonderen Lernleistung wird auf dem Abiturzeugnis auch dann zertifiziert, wenn der Schüler sie nicht in die Gesamtqualifikation einbringt.

#### **6. Anforderungen und Bewertung**

##### **6.1. Fachpraktischer Beitrag, schriftliche Arbeit, Kolloquium**

Besondere Lernleistungen können resultieren aus:

- umfassenden Beiträgen aus vom Freistaat Sachsen geförderten Wettbewerben
- Jahresarbeiten
- der Aufarbeitung umfassender, auch fachübergreifender Projekte oder Praktika.

Dabei ist die Erarbeitung fachpraktischer Beiträge (z. B. künstlerische Arbeiten, Entwicklung von Medien und Funktionsmodellen, Aufgabenlösungen in Leistungswettbewerben) denkbar.

Bedingungen für die Anerkennung einer Arbeit als Besondere Lernleistung sind gezielte Aufarbeitung und systematische Reflexion von Arbeitsgegenstand, Arbeitsverlauf und Arbeitsergebnis. Diese Forderung gilt ausnahmslos für alle Themen.

Wesentlicher Bestandteil der Besonderen Lernleistung ist in jedem Fall eine schriftliche Dokumentation, die einem Kolloquium zu präsentieren ist. Die schriftliche Dokumentation, deren Umfang etwa 15 Seiten betragen sollte, wird in ansprechender äußerer Form vorgelegt. Dazu gehören eine saubere und übersichtliche Form ebenso wie eine wirksame und präzise optische Gestaltung.

Die Dokumentation enthält z.B.

- in der Einleitung: die Erläuterung und Abgrenzung des (möglichst originellen) Themas, die Begründung seiner Relevanz;
- im Hauptteil: den Nachweis der Verwendung angemessener Methoden, das geeignete Fixieren und die übersichtliche Darstellung der Ergebnisse sowie ggf. deren kritische Diskussion, eine kritische Methodenreflektion;
- im Schlußteil: die Darstellung möglicher Konsequenzen, Querverbindungen, Anwendungen und Auswirkungen.

Ein normgerechtes Quellenverzeichnis muss und ein möglicher Anhang sowie eine Kurzfassung sollten die Arbeit abschließen.

Das abschließende, öffentliche Kolloquium dient der Präsentation des Arbeitsergebnisses.

Die Schüler bzw. Schülergruppen weisen sich als Autoren der Arbeit mit fundierten Kenntnissen zu Zielen, Methoden, inhaltlichen Details und Ergebnissen aus.

## **6.2. Leistungsbeurteilung, Leistungsermittlung**

Das Bewertungsverfahren ist vor Beginn der Arbeit zu besprechen.

Bewertungsgrundlage für die schriftliche Dokumentation ist der Nachweis der Beherrschung von Methoden, die auf wissenschaftliche Arbeitsweise vorbereiten. Dazu gehören:

- Wert und Umfang der Argumente;
- Konzentration auf das Wesentliche, Komprimiertheit; Präzision und logische Nachvollziehbarkeit der Darstellung
- Benennen der Gültigkeitsbedingungen der Ergebnisse
- Reflexion der Methoden und Lösungen - insbesondere bei mehreren möglichen Varianten;
- Originalität, Kreativität, Selbständigkeit und Problemorientierung;
- Qualität und Umfang der Recherchen;
- exakte Dokumentation des Arbeitsprozesses;
- Nachweis der Arbeitskontakte und Kooperationspartner.

Die schriftliche Dokumentation der Besonderen Lernleistung wird vom betreuenden Lehrer bewertet. Eine Zweitkorrektur, die auch durch externe Experten vorgenommen werden kann, ist in jedem Falle wegen der Bedeutung der Ergebnisse für das Abitur notwendig.

Wurde die Arbeit extern betreut erfolgt die Erstkorrektur durch den externen Kooperationspartner. Aus rechtlichen und organisatorischen Gründen ist die Mitwirkung des betreuenden Lehrers bei der Beurteilung und Ermittlung der Leistungen unverzichtbar. Er übernimmt dann die Funktion des Zweitkorrektors.

Bewertungsgrundlagen des Kolloquiums sind:

- Umfang des Wissens und Könnens;
- Argumentationssicherheit;
- Konzentration, Logik, Verständlichkeit der Ausführungen; Reaktionsfähigkeit, Engagement, Rhetorik;
- Sicherheit und Schauwert der Präsentation, wie z.B. fachpraktischer Vorführungen.

Die Bewertung des Kolloquiums erfolgt durch eine Prüfungskommission des betreffenden Gymnasiums, die analog einer Fachprüfungskommission der mündlichen Abiturprüfung zusammengesetzt ist. Mitglied ist in jedem Falle der Betreuer. Auch hier können sich betreuende Fachlehrer durch externe Fachleute beraten und unterstützen lassen. Wird die Arbeit von einem Externen betreut, stellt die Schule für das Kolloquium einen zusätzlichen Schriftführer ohne Stimmrecht.

Die Gewichtung der mündlichen Leistung im Kolloquium zur schriftlichen Arbeit erfolgt im Verhältnis 1: 2.

Sofern die Besondere Lernleistung eine fachpraktische Komponente enthält, gilt die Gewichtung: fachpraktische Komponente zu schriftliche Arbeit zu Kolloquium wie 1:1:1.

Fachpraktische Komponenten sind z.B.:

- durch zuständige Fachlehrer in ursächlichem Zusammenhang mit der Besonderen Lernleistung bewertete Arbeitsergebnisse aus Projekten, Praktika oder künstlerischer Tätigkeit;
- durch die jeweiligen Fachjurs bewertete Wettbewerbsbeiträge.

Bei Wettbewerben ist eine Einzelfallprüfung sowohl hinsichtlich einzelner Anforderungskriterien als auch hinsichtlich der Gesamtwettbewerbsleistung unverzichtbar. Die Begleitung der Schüler durch Lehrer des betreffenden Gymnasiums ist aus juristischen, organisatorischen und sachlichen Gründen notwendig. Sie müssen vor allem den zeitlichen Umfang, den individuell und selbständig erbrachten Anteil sowie die Verbindung zu schulischen Referenzfächern einschätzen bzw. bestätigen und in die abschließende Bewertung einbringen können.

Bei Gruppenarbeit gilt folgender Bewertungsmodus:

Die gemeinsam erbrachte Leistung wird mit bis zu 15 Punkten der einfachen Wertung bewertet (Grundbewertung). Diese Punktzahl wird mit der Anzahl der Gruppenmitglieder multipliziert und dadurch ein Punktepool gebildet. Dieser Pool wird auf die Mitglieder entsprechend ihrem individuellen Beitrag (ggf. zwischen den Extremen 0 Punkte und 15 Punkte) so aufgeteilt, dass der Pool exakt ausgeschöpft wird.

Die Leistung der einzelnen Schüler muss individualisierbar sein. Eine Gruppe soll deshalb nicht mehr als drei Mitglieder umfassen.

Mit Hilfe einer Checkliste kontrolliert und bestätigt der betreuende Lehrer die schrittweise Erfüllung der Anforderungen einer Besonderen Lernleistung.

Thomas Weise  
Tschaikowskistraße 30  
09130 Chemnitz  
Klasse 11/6  
Tutorgruppe Herr Schmidt  
Donnerstag, 4. Februar 1999

Hiermit melde ich mein Projekt „TS Formula“ als besondere Lernleistung an, bei der ich von Herrn Gerd Kunert (TU Chemnitz) und Herrn König betreut werde.

Bei dem Projekt handelt es sich um die Entwicklung eines Computerprogramms zum schnellen Auffinden und selbständigem Lösen von physikalischen, mathematischen und chemischen Formeln und Problemen.

Thomas Weise