

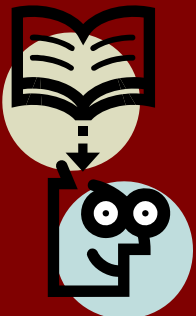
Manfred Becker

über  
100  
Seiten!

# Casio Basic Tutorial 1.00

© 05.09.2009

Basic-Programmierung  
für Anfänger  
mit den Taschenrechner  
**CASIO FX-850P/880P**



eBook

## Copyright

Alle in diesem Tutorial enthaltenen Programme und Verfahren wurden nach bestem Wissen erstellt und mit großer Sorgfalt getestet. Dennoch sind Fehler nicht ganz auszuschließen. Aus diesem Grund ist das im vorliegenden Tutorial enthaltene Programm-Material mit keiner Verpflichtung oder Garantie irgendeiner Art verbunden. Der Autor übernimmt infolgedessen keine Verantwortung und wird keine daraus folgende oder sonstige Haftung übernehmen, die auf irgendeine Art aus der Benutzung dieses Programm-Materials oder Teilen davon entsteht.

Die Wiedergabe von Gebrauchsnamen, Handelsnamen, Warenbezeichnungen usw. in diesem Werk berechtigt auch ohne besondere Kennzeichnung nicht zur Annahme, dass solche Namen im Sinne der Warenzeichen- und Markenschutz-Gesetzgebung als frei zu betrachten wären und daher von jedermann benutzt werden dürften.

Dieses Werk ist urheberrechtlich geschützt. Alle Rechte, auch die der Übersetzung, des Nachdrucks und der Vervielfältigung des Tutorials oder Teilen daraus, vorbehalten.

Autor:

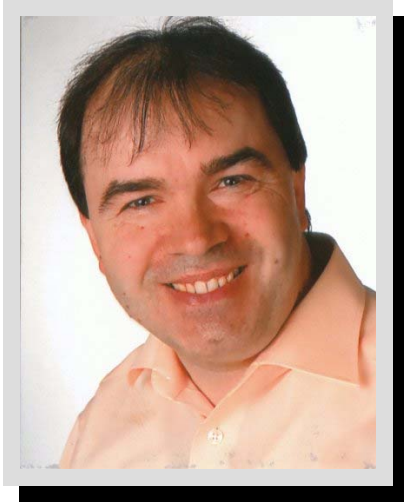
Dipl.-Ing.(FH) Manfred Becker

*Manfred Becker*

[mani.becker@web.de](mailto:mani.becker@web.de)

<http://manib.ma.funpic.de/>

## Vorwort



Ich hatte schon immer viel Freude dabei anderen zu helfen. Gerade bei Programmierfragen helfen mein fundiertes Wissen und meine große Erfahrung dabei, andere Hilfestellung zu geben. Bei einigen Foren habe ich das bereits erfolgreich praktiziert.

Allerdings habe ich festgestellt, dass meine Hinweise nur sehr wenigen Nutzern zugute kamen. Oft wurde ein und dieselbe Frage kurze Zeit später von einem Anderen erneut gestellt.

Besser ist es, so finde ich, ein ordentliches Tutorial anbieten zu können, in dem die Standardfragen abgehandelt werden. Nun genügt als Forenantwort der einfache Link in das entsprechende Kapitel.

Aus diesem Grund ist dieses Tutorial entstanden. Als Thema habe ich die **Casio Basic-Programmierung** herausgegriffen, da genau das auf dem Wunschzettel der meisten Anfänger ganz oben steht.

Ich hoffe es dient recht vielen Programmierern, und Alle die es werden wollen, bei ihrer täglichen Arbeit.

## In eigener Sache

Wenn Sie Vorschläge oder Fragen zum Tutorial haben, erreichen Sie mich per E-Mail:

[mani.becker@web.de](mailto:mani.becker@web.de)

oder im Forum auf meiner Homepage:

<http://manib.ma.funpic.de/>

Hier finden Sie auch die aktuelle Version und die zugehörnden BASIC-Quelltexte.

Aber auch wenn Sie keinen solchen Taschenrechner besitzen und dennoch wissen wollen wie man programmiert, finden Sie auf meiner Seite viele weitere sehr interessante Artikel. Schauen Sie doch einfach einmal vorbei. Ich freue mich über Ihren Besuch...

Manfred Becker  
Apr. 2008 bis Aug. 2009



# Inhaltsverzeichnis

1	Einführung .....	9
1.1	Casio FX-850P / FX-880P .....	9
1.2	Spezifikation Casio FX-850P / FX-880P .....	10
1.3	Preise und Zubehör für Casio Pocket Computer .....	11
2	Schnelleinstieg .....	12
2.1	Mein erstes Casio-Basic Programm .....	12
3	Voraussetzungen .....	13
3.1	Der Taschenrechner Casio FX-850P/FX-880P .....	13
3.1.1	CAL-Modus .....	14
3.1.2	Formelspeicher-Funktion .....	15
3.1.3	Datenbank-Funktion .....	16
3.1.4	Eingebaute Wissenschaftliche Bibliothek .....	17
3.1.5	BASIC-Modus .....	18
3.2	Die CASIO-Bedienungsanleitung .....	19
3.3	Interesse und Zeit .....	20
4	Grundlagen .....	21
4.1	Was ist Programmieren? .....	21
4.2	Was ist Basic? .....	22
4.3	Die Syntax des Casio-BASIC .....	23
4.3.1	Schlüsselwörter .....	23
4.3.2	Befehle .....	23
4.3.3	Anweisungen .....	23
4.3.4	Funktionen .....	23
4.3.5	Variablen .....	24
4.3.6	Zeilenformat .....	24
4.4	Der Wortschatz des Casio-BASIC .....	26
4.4.1	Manuelle Befehle .....	26
4.4.2	Grundbefehle .....	26
4.4.3	Numerische Funktionen .....	26
4.4.4	Zeichenfolgen-Funktionen .....	26
4.4.5	Eingabe/Ausgabe-Befehle .....	26
4.4.6	Datenbank-Befehle .....	26
4.5	Die Betriebsarten .....	27
4.5.1	Direktmodus .....	27
4.5.2	Indirekter Modus .....	27
4.6	Wie speichert bzw. - löscht man Programme? .....	27
4.7	Wie überträgt man Programme vom bzw. zum PC? .....	28
4.7.1	Daten zum Casio senden .....	28
4.7.2	Daten vom Casio empfangen .....	29
5	Mit Basic rechnen .....	31
5.1	Die Ausgabe von Zahlen .....	31
5.2	Wissenschaftliche Schreibweise .....	31
5.3	Mathematische Regeln .....	32
5.4	Formatierte Ausgabe .....	33
5.5	Übungen .....	34
5.5.1	Übung 5-1 .....	34
5.5.2	Übung 5-2 .....	34
5.5.3	Übung 5-3 .....	34
5.5.4	Übung 5-4 .....	34
5.5.5	Übung 5-5 .....	34

5.5.6	Übung 5-6 .....	34
5.5.7	Übung 5-7 .....	34
5.5.8	Übung 5-8 .....	34
5.5.9	Übung 5-9 .....	34
6	Variable speichern Zahlen oder Texte .....	35
6.1	Zwei Arten von Variablen .....	35
6.1.1	Numerische Variable.....	36
6.1.2	Zeichenfolge-Variable .....	36
6.2	Der INPUT-Befehl.....	38
6.2.1	Eingabe von numerischen Variablen.....	38
6.2.2	Eingabe von Zeichenfolgen-Variablen .....	38
6.2.3	Verwendung mehrerer Variablen bei einem INPUT-Befehl .....	38
6.3	Feldvariable .....	39
6.3.1	Eindimensionale Felder .....	39
6.3.2	Mehrdimensionale Felder.....	39
6.4	Konstante .....	40
6.5	Wertzuweisungen.....	40
6.6	Variable verändern .....	40
6.7	Übungen.....	41
6.7.1	Übung 6-1 .....	41
6.7.2	Übung 6-2 .....	41
6.7.3	Übung 6-3 .....	41
6.7.4	Übung 6-4 .....	41
6.7.5	Übung 6-5 .....	41
6.7.6	Übung 6-6 .....	41
6.7.7	Übung 6-7 .....	41
7	Tipps für das Programmieren (Teil 1) .....	42
7.1	Der REM-Befehl .....	42
7.2	Mehrere Anweisungen in einer Zeile.....	42
7.3	Gebrauch von Leerzeichen.....	43
7.4	Auswahl von Variablennamen .....	43
7.5	Richtige Zeilennummerierung .....	44
7.6	Bildschirm löschen und positionsgenaue Ausgabe .....	44
7.7	Weitere Eingabebefehle .....	45
7.7.1	Die INPUT\$ Funktion .....	45
7.7.2	Die INKEY\$ Funktion .....	45
8	Entscheidungen treffen.....	46
8.1	Die GOTO-Anweisung .....	46
8.2	Logische Ausdrücke.....	46
8.2.1	Vergleichsoperatoren.....	47
8.2.2	Logische Operatoren.....	48
8.3	Die IF-Anweisung.....	49
8.3.1	IF ~ THEN.....	49
8.3.2	IF ~ THEN ~ ELSE.....	49
8.3.3	IF ~ GOTO .....	50
8.3.4	IF ~ GOTO ~ ELSE .....	50
8.4	Ausführbare Befehle der IF-Anweisung.....	51
8.4.1	Anweisung .....	51
8.4.2	Verzweigungsziel .....	51
8.5	Eine Rechenübung .....	52
8.5.1	Einführung .....	52
8.5.2	Neue Funktionen für die Rechenübung .....	52
8.5.3	Programmversion 1.00 .....	53

8.5.4	Eine verbesserte Version 1.01.....	57
8.5.5	Eine neuer Ansatz Version 2.00.....	60
8.5.6	Eine verbesserte Version 2.01.....	61
8.6	Übungen.....	62
8.6.1	Übung 8-1 .....	62
8.6.2	Übung 8-2 .....	62
8.6.3	Übung 8-3 .....	62
8.6.4	Übung 8-4 .....	62
8.6.5	Übung 8-5 .....	62
8.6.6	Übung 8-6 .....	62
8.6.7	Übung 8-7 .....	62
9	Anweisungen wiederholen.....	64
9.1	Das IF / GOTO-Verfahren .....	64
9.2	Die FOR ... NEXT Schleife.....	64
9.3	Summe der ersten N ganzen Zahlen .....	65
9.4	Wertetabellen.....	65
9.5	Fortgeschrittene Schleifenstrukturen.....	66
9.5.1	Variable Schrittweite .....	66
9.5.2	Verschachtelte Schleifen .....	67
9.6	Übungen.....	67
9.6.1	Übung 9-1 .....	67
9.6.2	Übung 9-2 .....	67
9.6.3	Übung 9-3 .....	67
9.6.4	Übung 9-4 .....	67
9.6.5	Übung 9-5 .....	67
9.6.6	Übung 9-6 .....	67
9.6.7	Übung 9-7 .....	67
9.6.8	Übung 9-8 .....	67
9.6.9	Übung 9-9 .....	67
10	Tipps für das Programmieren (Teil 2) .....	68
10.1	Weitere Grundbefehle .....	68
10.2	Weitere Numerische Funktionen .....	68
10.3	Weitere Zeichen-Funktionen .....	68
10.4	Weitere Eingabe/Ausgabe-Befehle .....	68
10.5	Die Datenbank-Funktion nutzen .....	68
10.6	Die eingebaute Wissenschaftliche Bibliothek nutzen .....	68
10.7	Fehlerbehandlung .....	68
10.8	Debugging .....	68
11	Beispielprogramme .....	69
11.1	Zahlenraten .....	70
11.2	Code-Knacker.....	71
11.3	Hangman.....	73
11.4	TRANSLIB .....	75
11.5	xxxx .....	76
11.6	xxxx .....	76
11.7	xxxx .....	76
11.8	xxxx .....	76
11.9	xxxx .....	76
11.10	xxxx .....	76
12	Anhang.....	78
12.1	Antworten zu den Übungen .....	78
12.1.1	Antworten zu Übungen aus Kapitel 5.....	78
12.1.2	Antworten zu Übungen aus Kapitel 6.....	78

12.1.3	Antworten zu Übungen aus Kapitel 8.....	79
12.1.4	Antworten zu Übungen aus Kapitel 9.....	80
12.2	Begriffserklärung .....	81
12.2.1	Interpreter.....	81
12.2.2	Compiler.....	81
12.2.3	Assembler.....	81
12.2.4	Object-Code .....	81
12.2.5	Library .....	81
12.2.6	Linker.....	82
12.3	Befehls-Übersicht.....	83
12.4	Übersicht der Wissenschaftlichen Bibliothek .....	87
12.5	Belegungsplan des Speichers .....	91
12.6	Belegungsplan des Erweiterungs-Steckers .....	92
12.7	Befehls-Referenz.....	93
12.7.1	A .....	93
12.7.2	B .....	93
12.7.3	C .....	93
12.7.4	D.....	93
12.7.5	E .....	93
12.7.6	F .....	94
12.7.7	G.....	94
12.7.8	H.....	94
12.7.9	I.....	94
12.7.10	L .....	95
12.7.11	M.....	95
12.7.12	N.....	95
12.7.13	O .....	95
12.7.14	P .....	95
12.7.15	R .....	96
12.7.16	S .....	96
12.7.17	T .....	96
12.7.18	V .....	97
12.7.19	W .....	97
12.8	Index .....	98

# 1 Einführung

## 1.1 Casio FX-850P / FX-880P

Meinen Casio FX-850P habe ich inzwischen schon sehr lange (seit 1991). Ich habe ihn damals für mein Studium gekauft. Allerdings musste ich feststellen, dass er dafür nur zum Teil geeignet war. Er musste deshalb in der Mathe-Vorlesung einem HP 48SX weichen. Der HP war für Integral-Rechnungen und für Gleichungslösungen einfach unschlagbar.



Abbildung 1

Dennoch blieb ich dem Casio treu, nicht zuletzt deswegen, weil er sehr unkompliziert ist. Die Basic-Programmierung ist einfach genial, und ich konnte ihn in manch anderen Vorlesungen verwenden.

Ein weiteres Phänomen ist die große Beliebtheit dieses Rechners, und das obwohl er längst ein Museumsstück sein könnte. Er wird zwar schon länger nicht mehr hergestellt, aber nichts desto trotz steht er (und sein Bruder FX-880P) auch heute noch hoch im Kurs. Bei eBay Auktionen werden Preise bis zu 200.- Euro geboten!

Aufgrund des Alters findet man aber heute leider kaum noch Zubehör oder Dokumentation zu dem Rechner. Bücher sind ebenfalls schwer zu bekommen. Auch im Internet werden viele der Webseiten, die Themen zum Casio anbieten, kaum noch gepflegt. Klar, es gibt Ausnahmen, doch wenn man auf der Suche nach Informationen ist, hat man es heutzutage schwer.

Das hat mich dazu bewogen meine alten Daten zu diesem Rechner auszugraben, und auf meiner Homepage online zu stellen. Ich hoffe, die Artikel machen ihnen auch heute noch viel Freude.

Ein Thema hatte aber bis jetzt gefehlt: **Eine Einführung in die Casio-Basic Programmierung!**

Das will ich nun mit diesem Tutorial nachholen. Ich habe mich dabei zurückversetzt in meine Programmieranfänge. Alle Fragen und Probleme die ich damals hatte, will ich versuchen hier aufzuzeigen, um ihnen dann die Lösung präsentieren zu können. Dabei will ich das Wichtigste herausstellen, ins Detail gehe ich dann, wenn es notwendig wird. Dadurch, so hoffe ich, gelingt es ihnen mit mir zusammen durch all die Unwägbarkeiten zu kommen, um am Ende sagen zu können: „jetzt weiß ich, wie so was funktioniert, wie man an die Sache anpackt, und wie man das umsetzen kann“.

In diesem Sinne wünsche ich Ihnen beim Lesen dieser Lektüre genauso viel Spaß, wie ich beim Schreiben hatte...

## 1.2 Spezifikation Casio FX-850P / FX-880P

Hier zunächst einmal die Technischen Daten des Casio FX-850P / FX-880P.

- **Anzeige**  
Punktmatrix-Flüssigkristallanzeige mit 32 Spalten x 2 Zeilen (5x7 Punkte)  
Symbol-Anzeigen und eine 5-stellige 7-Segment Anzeige  
LCD Ansteuerungs-IC: 2 x HD66100F  
Regelbarer Kontrast
- **Prozessor**  
CPU: VLSI mit 1,228MHz. Hitachi HD62002A01  
(FX-860P, FX-880P: HD62002A03)
- **Speicher**  
8 KB RAM (FX-860P: 24 KB, FX-880P: 32 KB)  
Interner Einbauschacht für Speichererweiterungskarte RP-8 bzw. RP-33  
Maximale Speicheraufrüstung insgesamt 40 KB (FX-880P: 64 KB)
- **Spannungsversorgung**  
2x 3V CR2032 Lithium Batterien für Haupt-Spannungsversorgung  
1x 3V CR1220 Lithium Batterie für Speicherschutz-Spannungsversorgung  
Leistungsaufnahme 0.04 W  
Ausschaltautomatik nach ca. 6 Minuten
- **Batterie-Lebensdauer**  
ca. 90 Stunden bei Dauerbetrieb  
ca. 150 Stunden bei Daueranzeige von 5555555555  
ca. 4.5 Monate bei täglich einstündiger Benutzung
- **Programmiersprache**  
BASIC, 10 Programmbereiche (P0 bis P9)
- **Eingebauter Lautsprecher**
- **Anschlussstecker für Peripheriegeräte**  
Unterstützt RS232 und Centronics  
(Spezielles Übertragungskabel oder Interfaceeinheit FA-6 notwendig )
- **Abmessungen**  
11.6 x 193 x 78 (HxBxT) mm
- **Gewicht**  
197g mit Batterien
- **Zubehör**  
Etui, Referenzkarte und Bedienungsanleitung

Der Taschenrechner hat einen BASIC Interpreter, eine MEMO Funktion, eine Bibliothek mit 116 wissenschaftlichen Funktionen, und eine Schnittstelle, die es erlaubt zu anderen Geräten Verbindung aufzunehmen. Der Speicher kann insgesamt auf 40 KB aufgerüstet werden, wenn man die Speichererweiterungskarte RP-33 einsetzt (8 KB + 32 KB).

Später wurde der Casio FX-880P gebaut, der dem FX-850P gleicht, außer dass bei ihm standardmäßig 32 KB interner Speicher eingebaut waren. Somit kann der FX-880P auf insgesamt 64 KB aufgerüstet werden.

Die Bedienungsanleitung können Sie im Internet herunterladen:

- <http://www.silrun.info>
- <http://www.usersmanualguide.com/casio/calculators/fx-850p>

### 1.3 Preise und Zubehör für Casio Pocket Computer

Folgende CASIO Pocket Computer nebst Zubehör kann (bzw. konnte) man erwerben.

Pocket Computer:

• FX-730P	8 KB RAM	159,00 DM
• FX-795P	16 KB RAM	229,00 DM
• FX-850P	8 KB RAM	249,00 DM
• FX-880P	32 KB-RAM	299,00 DM

Zubehör:

• FA-6	RS-232 + Kassetteninterface	299,00 DM
• SB-7	Kassettenrecorder Kabel	39,90 DM
• PK-7	Kabel FP-100, SB-43, FA-6	79,90 DM
• RP-8	8 KB-RAM Erweiterung	49,90 DM
• RP-33	32 KB-RAM Erweiterung	99,90 DM
• FP-40	Plotter 40/80 Zeichen	249,00 DM
• SB-43	RS-232 Centronicsbox an FP-40	89,90 DM
• SB-51	Centronicskabel FA-80 -> FP-40	149,00 DM

Wenn Sie heutzutage ein Übertragungskabel suchen, müssen Sie schon viel Glück haben, eines zu ergattern. Selbst bei eBay werden nur sehr selten solche Kabel angeboten. Und diese Kabel benötigen dann fast immer eine Serielle Schnittstelle. Doch diese ist bei neuen Rechnern und Notebooks durch eine USB-Schnittstelle ersetzt.

Aus diesem Grund habe ich ein USB-Interface entwickelt, welches Sie über meine Homepage beziehen können!



Abbildung 2

• USB-Interface	39,90 €
-----------------	---------

<http://manib.ma.funpic.de/>

## 2 Schnelleinstieg

In diesem Kapitel will ich für alle Ungeduldigen unter ihnen eine Kurzeinweisung geben, wie Sie ein Basic Programm erstellen und dann ausführen können.

### 2.1 Mein erstes Casio-Basic Programm

Nach dem ersten Anschalten des Rechners, sollten Sie nebenstehende Anzeige sehen.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
-
```

Der Casio ist im CAL-Modus, in dem manuelle Berechnungen durchgeführt werden können.

Um ein Basic-Programm eingeben und starten zu können, muss man in den BASIC-Modus umschalten. Drücken Sie dazu nacheinander die Tasten **MODE** und **1**.

Beachten Sie die Modus-Anzeige, die von **CAL** auf **BASIC** umgesprungen ist.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
P 0 1 2 3 4 5 6 7 8 9      35368
Ready P0
```

Tippen Sie die nebenstehende Programmzeile **10** ein, und drücken Sie zum Abschluss die **EXE** Taste.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
Ready P0
10 PRINT "Hallo Casio! ";
```

Tippen Sie die nebenstehende Programmzeile **20** ein, und drücken Sie zum Abschluss die **EXE** Taste.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINT "Hallo Casio! ";
20 GOTO 10
```

Tippen Sie den Befehl **RUN** ein, und drücken Sie zum Abschluss die **EXE** Taste.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
20 GOTO 10
RUN
```

Jetzt sollte in der Casio-Anzeige der Text "Hallo Casio!" durchlaufend angezeigt werden.

Mit der **BRK** Taste machen Sie dem Spuk ein Ende!

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
Break
Ready P0
```

Das Programm bleibt auch noch nach dem Ausschalten erhalten. Sie können es beliebig oft starten und beenden.

Ich hoffe Sie sind etwas Beeindruckt, wie einfach es ist, mittels BASIC Ihrem Taschenrechner „Leben“ einzuhauchen. Sie müssen jetzt noch nicht verstanden haben, was genau Sie da eben eingetippt haben (obwohl es nicht wirklich schwer ist). Das werde ich in den folgenden Kapiteln noch im Detail erklären...

### 3 Voraussetzungen

Um mit diesem Tutorial erfolgreich arbeiten zu können, sollten Sie folgende Dinge haben:

- Den Taschenrechner Casio FX-850P/FX-880P
- Die dazugehörige CASIO-Bedienungsanleitung
- Dieses Tutorial, Interesse und Zeit

#### 3.1 Der Taschenrechner Casio FX-850P/FX-880P

Ja, das ist er. Ich gebe zu, ich habe schon immer ein Fabel für alles Technische gehabt. Insbesondere Taschenrechner haben es mir angetan. Aber dieser Casio ist etwas ganz besonderes.



Abbildung 3

Er ist viel mehr wie ein normaler Taschenrechner. Er hat fünf Betriebsarten, ein sehr gutes, ausführliches Handbuch, und ein schönes, schlankes Design. Das Alu-Gehäuse, das große Display und die QWERTY-Tastatur verleihen dem Rechner etwas Erhabenes. Dann die abgesetzte Nummerntastatur mit den typischen Taschenrechner-Funktionen an der Seite rundet den positiven Eindruck ab. Die Tastatur (Plastiktasten) ist sehr gut zu bedienen. Sie hat zwar keinen so guten Druckpunkt, wie etwa die von einem HP 48SX Rechner, aber ist immer noch besser wie die Gummi-Tastatur von einem SHARP PC-E500(S).

Ehrlich gesagt, gibt es nur drei Dinge, die man an ihm bemängeln könnte:

- Er besitzt leider kein grafisches Display, sondern „nur“ eine 2x32 Zeichen Anzeige.
- In seiner Grundausstattung hat der FX-850P lediglich 8 kByte RAM (der FX-880P hat übrigens standardmäßig 32kByte)
- Er wird nicht mehr hergestellt

Aber das mit der Grafik ist nicht ganz so schlimm, und den Speicher kann man nachträglich aufrüsten. Schade nur, dass er nicht mehr hergestellt wird. Lediglich bei Flohmärkten oder bei eBay kann man das gute Stück manchmal noch ergattern.

Aber schauen wir uns die fünf Betriebsarten (Betriebsmodi oder kurz Modi) an.

Die Betriebsarten des Casio FX-850P/FX-880P:

- CAL-Modus
- Formelspeicher-Funktion
- Datenbank-Funktion
- Eingebaute Wissenschaftliche Bibliothek
- BASIC-Modus

### 3.1.1 CAL-Modus

Zunächst einmal ist da der CAL-Modus, mit dem man ihn wie ein ganz normaler wissenschaftlicher Taschenrechner bedienen kann. Von großem Vorteil ist natürlich das für einen Taschenrechner ungewöhnliche Display, mit dem 2 Zeilen zu je 32 Zeichen dargestellt werden können. Über diesen zwei Zeilen befinden sich die Symbolanzeigen, welche die verschiedenen Betriebsmodi anzeigen.

Schön ist auch, dass über dem Ziffernblock die Tasten für die wissenschaftlichen Funktionen angeordnet sind. Das ist Standard bei vielen Wissenschaftlichen Taschenrechnern, und man muss sich nicht umgewöhnen.

Nach dem Einschalten befindet sich der Rechner automatisch im CAL-Modus, in dem arithmetische Berechnungen, Berechnungen mit Funktionen, Ausführung der wissenschaftlichen Bibliothek, Formelspeicher-Berechnungen, Programmausführung und Datenabruf durchgeführt werden können.



Die Bedienung ist wie bei normalen Taschenrechnern. Allerdings wird das Ergebnis in der zweiten Zeile angezeigt. Dadurch hat man Berechnung und Ergebnis immer gleichzeitig auf dem Display, was sicherlich von Vorteil ist.

Anstelle der Taste **=** muss die Taste **EXE** verwendet werden.



Ein weiterer Vorteil dieses Taschenrechners ist die Verwendung von Variablen im CAL-Modus. Was Variable sind werde ich später noch im Detail erklären. Hier nur soviel: Einer Variablen kann man Werte zuweisen, die der Rechner sich merkt, und erst beim Ausschalten wieder vergisst. Die Variable wird über ihren Variablennamen, angesprochen. Dieser Variablenname kann frei gewählt werden. Er muss allerdings mit einem Buchstaben beginnen, und darf keine Sonderzeichen enthalten (später mehr).

Z.B. wird hier der Variablen X der Wert 4 zugewiesen.



Danach kann eine Berechnung mit dieser Variablen durchgeführt werden.






**Hinweis**

Die Bedienung und Berechnungsfunktionen sind in der CASIO-Bedienungsanleitung, Kapitel 3 ab Seite 15 ausführlich beschrieben.


### 3.1.2 Formelspeicher-Funktion

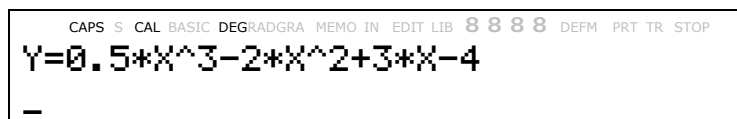
Ein echtes Highlight ist wohl die Formelspeicher-Funktion. Damit lassen sich Berechnungen, die sich mehrfach wiederholen, ganz einfach durchführen. Ideal also, wenn es darum geht Wertetabellen zu einer Formel zu erstellen.

Folgende drei Tasten spielen dabei eine Rolle:

-  Speichert die angezeigte Formel
-  Zeigt die gespeicherte Formel an
-  Führt die Berechnung der Formel aus

Zunächst einmal muss die gewünschte Formel eingegeben werden.


Nach der Eingabe muss die Taste  gedrückt werden.




Wir nehmen einmal an, dass die Wertetabelle der Funktion von 0 bis 9 zu berechnen wäre. Nun kann mit der Taste  die Berechnung der Formel gestartet werden.

Der Rechner fordert zur Eingabe der Variablen X auf.



Nach der Eingabe der 0 und drücken der Taste , wird das Ergebnis angezeigt.





Nach erneuten drücken der  Taste wird die nächste Eingabe erwartet.



Dieser Vorgang kann beliebig oft wiederholt werden.



Die Taste  unterbricht die Berechnung. Später kann man mit der Taste  wieder weitermachen.



Wertetabelle zu:  $f(x) = 0.5x^3 - 2x^2 + 3x - 4$

x	0	1	2	3	4	5	6	7	8	9
f(x)	-4	-2.5	-2	0.5	8	23.5	50	90.5	148	225.5



Hinweis

Die Formelspeicher-Funktion ist in der CASIO-Bedienungsanleitung, Kapitel 4 ab Seite 33 beschrieben.

### 3.1.3 Datenbank-Funktion





Die Datenbank-Funktion ist ein weiteres Feature, das dieser Taschenrechner bietet. Damit kann man alles Mögliche eingeben und verwalten. Es kann als Notizbuch, Adressspeicher, Telefonliste, Formelspeicher, (Spickzettel;) u.v.m eingesetzt werden.

Die eingegebenen Daten können später erweitert, editiert oder wieder gelöscht werden. Sehr interessant ist die Tatsache, dass man auf diese Daten auch von einem Basic-Programm zugreifen kann. Außerdem lassen sich abgelegt Formeln ganz einfach mit der Formelspeicher-Funktion verwenden.

Die Dateneingabe erfolgt im MEMO-IN-Modus. Dieser wird durch die Betätigung der Tasten **MODE** **9** aktiviert.

Im MEMO-IN-Modus lassen sich Daten ganz einfach eingeben.



Die gespeicherten Daten können nun angezeigt werden. Dazu ist die Taste **MEMO** zu drücken. Hierbei kann man mit den Cursor-Tasten     zwischen den einzelnen Datensätzen wechseln.

Sollen weitere Daten aufgenommen oder vorhandene Daten editiert werden, muss der MEMO-IN-Modus aktiviert werden. Auch das Löschen von Datensätzen kann hierbei durchgeführt werden. Verlassen wird der MEMO-IN-Modus indem man einen anderen Modus aktiviert. Z.B. den CAL-Modus mit **MODE** **0**.

Mann kann auch nach einem Begriff suchen. Dazu tippt man den gesuchten Begriff im CAL-Modus ein, und drückt dann die **MEMO** Taste. Sollte der Begriff in den Daten vorkommen, wird der entsprechende Datensatz sofort angezeigt. Mit jedem folgenden Drücken von **MEMO** wird das nächste Datenelement angezeigt, das mit dem Suchbegriff übereinstimmt.

Die Datenbank-Daten können aber auch in einem Basic-Programm verwendet werden. Dazu stehen folgende Basic-Befehle zur Verfügung:

READ#	Liest Daten aus der Datenbank
RESTORE#	Positioniert den Lesezeiger
WRITE#	Schreibt Daten in die Datenbank



#### Hinweis

Die Datenbank-Funktion ist in der CASIO-Bedienungsanleitung, Kapitel 5 ab Seite 37 ausführlich beschrieben.

### 3.1.4 Eingebaute Wissenschaftliche Bibliothek

Bei diesem Taschenrechner kann auf eine Bibliothek von insgesamt 116 Programmen zurückgegriffen werden. Diese Programme behandeln mathematische, statische sowie physikalische und wissenschaftliche Themen. Am Ende dieses Tutorials habe ich die Übersicht aller 116 Programme angefügt.

Die Programme sind im Handbuch vorbildlich beschrieben. Jedes der Programme hat eine vierstellige Bibliotheksnummer. Der Aufruf erfolgt dementsprechend durch die Eingabe dieser Nummer gefolgt von der **LIB** Taste.

z.B. Gerade durch zwei Punkte:  
5510 **LIB**

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
y=ax+b ←- (x1,y1),(x2,y2)
x1= 0 ?_
```

Aber noch Interessanter wird das Ganze, wenn man im Handbuch liest, dass alle Bibliotheksprogramme in BASIC programmiert sind! Und es kommt noch besser, denn die Programmlistings kann man durch einen Trick in den Basic-Programmspeicher laden. Dort kann man sich dann das Programmlisting anschauen und somit direkt von den Casio-Profis lernen.

Der Trick, wie sollte es auch anders sein, wird über ein Basic Programm realisiert. Dieses Basic-Programm (TRANSLIB) finden Sie im Kapitel zu den Beispielprogrammen. Ich habe mir für Sie die Mühe gemacht, alle 116 Bibliotheksprogramme auszulesen. Das Ergebnis finden Sie auf meiner Homepage. In meinem Casio Basic Library finden Sie alle Listings zum Scientific Library 116.

Hier sehen Sie z.B. das Listing zum oben gezeigten Bibliotheksprogramm 5510:

#### Listing 1 5510 Gerade durch zwei Punkte

```
5 ONERRORGOTO500
10 MODE8:DIM:ERASEa:DIMa(5):a$="x1y1x2y2ab":c$=CHR$(5)
20 CLS:PRINT"y=ax+b ←- (x1,y1),(x2,y2)";
30 FORi=0 TO3
40 LOCATE0,1:PRINTc$:MID$(a$,1+2*i,2)="":a(i):"?":INPUT@42:a(i):LOCATE0,0
50 NEXT:LOCATE0,1:PRINTc$:
70 IFA(0)=a(2) THENIFA(1)=a(3) THEN100 ELSEPRINTc$:"x =":a(0):w$=INPUT$(1,@)
:GOTO30
72 IFA(1)=a(3) THENPRINTc$:"y =":a(1):w$=INPUT$(1,@):GOTO30
75 a(4)=(a(3)-a(1))/(a(2)-a(0)):a(5)=a(1)-a(4)*a(0)
80 FORi=4 TO5:PRINTc$:MID$(a$,i+1,1):" =":a(i);
82 w=ASC(INPUT$(1,@)):IFw=13 THEN90
84 IFw=30 ANDi>4 THENi=i-2:GOTO90
85 IFw=31 ANDi<5 THEN90
88 GOTO82
90 NEXT:GOTO30
100 LOCATE0,1:PRINTc$:"not found":w$=INPUT$(1,@):GOTO20
500 IFERR=1 THENCLS:ONERRORGOTO0
510 IFERL=40 THENRESUME40
520 RESUME100
```



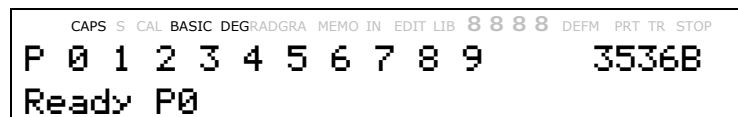
#### Hinweis

Die Eingebaute Wissenschaftliche Bibliothek ist in der CASIO-Bedienungsanleitung, Kapitel 11 ab Seite 180 ausführlich beschrieben.

### 3.1.5 BASIC-Modus

Der Basic-Modus ist das Beste am Casio Rechner. Im Gegensatz zu „normalen“ Taschenrechnern, bei denen die Funktionen fest vorgegeben und unveränderbar sind, kann man hier über die Basic-Programmierung die Funktionen beliebig erweitern und anpassen. Die Vorstellung einem Rechner sagen zu können, was er tun soll, hat mich fasziniert seit ich das erste Mal<sup>1</sup> einen Sinclair ZX-81 in den Fingern hatte. Und diese Faszination hat mich bis heute nicht wieder losgelassen.

Beim Casio dient der BASIC-Modus für die Erstellung und Ausführung der Programme. Dieser Modus wird durch das Drücken der Tasten **MODE** **1** aktiviert.



Es gibt 10 Programmbereiche (P0 bis P9) in denen jeweils ein eigenes Basic-Programm abgelegt werden kann. Dieser Programmbereich kann im Basic-Modus über die Tastenfolge **S** gefolgt von **0** bis **9** gewechselt werden. Programme bleiben nach der Eingabe erhalten. Man muss also nicht Sichern, wie man das z.B. von einem Editor-Programm auf einem PC gewöhnt ist.

Ein Basic-Programm besteht meist aus mehreren Programmzeilen, der Taschenrechner hat aber nur 2 Zeilen zur Anzeige zur Verfügung. Aus diesem Grund gibt es einen speziellen Edit-Modus, bei dem über die Cursor-Tasten **↑** **↓** **←** **→** durch das Programmlisting gescrollt werden kann.

Außerdem lassen sich im Basic-Modus manuelle Befehle direkt eingeben (wie z.B. LIST, EDIT, PRINT oder RUN). Eine Basic-Programmzeile dagegen beginnt immer mit einer Zeilennummer (1 bis 65535), gefolgt von einer oder mehreren Basic-Anweisungen.

Der Casio hat eine Vielzahl von Funktionen und Befehle, welche als Anweisungen verwendet werden können. Vor allen bei den Mathematischen Funktionen bietet das Casio Basic eine sehr große Auswahl.

Erstellte Programme werden mit dem Befehl RUN gestartet. Ein laufendes Programm kann mit der Taste **BRK** unterbrochen werden. Wenn ein Programm eines Programmbereichs gelöscht werden soll, verwendet man den Befehl NEW. Sollen alle Programme in allen Programmbereichen gelöscht werden, verwendet man NEW ALL. Einzelne Zeilen eines Programms werden durch die Eingabe der Zeilennummer, gefolgt von **EXE** gelöscht.



**Hinweis**

Die Basic-Programmierung ist in der CASIO-Bedienungsanleitung, Kapitel 6 ab Seite 46 ausführlich beschrieben.  
Die Basic-Befehls-Referenz finden Sie im Kapitel 10, ab Seite 91.

<sup>1</sup> Das war 1983, während meiner Ausbildungszeit.

### 3.2 Die CASIO-Bedienungsanleitung

Ich gehe davon aus, dass Sie im Besitz der CASIO-Bedienungsanleitung sind, da ich ab und zu auf dortige Kapitel verweise. Falls nicht, rate ich Ihnen die CASIO-Bedienungsanleitung im Internet<sup>2</sup> herunter zu laden. Meiner Meinung nach ist die CASIO-Bedienungsanleitung ein sehr gutes Nachschlagewerk. Es ist in folgende 11 Kapitel unterteilt:

1. Konfiguration des Geräts.....	1
2. Grundbedienung.....	11
3. Berechnungsfunktionen.....	15
4. Formelspeicher-Funktion.....	33
5. Datenbank-Funktion.....	37
6. Basic-Programmierung.....	45
7. Peripheriegeräte.....	71
8. Kompatibilität mit der Serie PB-100.....	80
9. Grundlegendes zum Umgang mit Dateien.....	87
10. Befehls-Referenz.....	91
11. Wissenschaftliche Bibliothek.....	180

Wie Sie sehen, wird alles Notwendige behandelt. Schade nur, dass bei der Befehlsreferenz nicht einfach nur nach dem Alphabet sortiert wurde. Das würde das Auffinden wesentlich vereinfachen. So muss man sich zunächst im Klaren sein, unter welcher Kategorie man suchen soll:

1. Manuelle Befehle.....	92
CLEAR, EDIT, FRE, LIST, LIST ALL, NEW, NEW ALL, PASS, RUN, TROFF, TRON, VARLIST	
2. Grundbefehle.....	100
BEEP, CLS, DATA, DEFSEG, DIM, END, ERASE, ERL, ERR, FOR ~ NEXT, GOSUB, GOTO, IF ~ THEN ~ ELSE, IF ~ GOTO ~ ELSE, INKEY\$, INPUT, INPUT\$, LET, LOCATE, ON ERROR GOTO, ON GOSUB, ON GOTO, PEEK, POKE, PRINT, READ, REM ( ' ), RESTORE, RESUME, RETURN, SET, STOP, TAB	
3. Numerische Funktionen.....	133
ABS, ACS, ANGLE, ASN, ATN, COS, CUR, EXP, FACT, FIX, FRAC, HYP ACS, HYP ASN, HYP ATN, HYP COS, HYP SIN, HYP TAN, INT, LN, LOG, NCR, NPR, PI, POL, RAN#, REC, ROUND, SGN, SIN, SQR, TAN	
4. Zeichen-Funktionen.....	150
&H, ASC, CHR\$, DEG, DMS\$, HEX\$, LEFT\$, LEN, MID\$, RIGHT\$, STR\$, VAL, VALF	
5. Eingabe/Ausgabe-Befehle.....	162
CLOSE, EOF, INPUT#, INPUT\$, LLIST, LOAD, LOAD ALL, LPRINT, OPEN, PRINT#, SAVE, SAVE ALL, VERIFY	
6. Datenbank-Befehle.....	174
LIST#, LLIST#, LOAD#, NEW#, READ#, RESTORE#, SAVE#, WRITE#	

Am Ende dieses Tutorials habe ich deshalb die alphabetisch geordnete Befehls-Übersicht angefügt, incl. Verweis auf die Seite im Casio Handbuch.

<sup>2</sup> <http://www.silrun.info/>

### 3.3 *Interesse und Zeit*

Eine wesentliche Voraussetzung für das erfolgreiche Bearbeiten dieses Tutorials ist natürlich ihr Interesse an diesem Thema. Gleichzeitig müssen Sie auch einiges an Zeit investieren. Um das Programmieren zu erlernen ist es leider nicht damit abgetan dieses Tutorial von vorne bis hinten durchzulesen. Viel wichtiger ist es, die Theorie in die Praxis umzusetzen. Und für dieses Erarbeiten benötigt man viel Zeit.

Gerade zu Beginn wird es häufig vorkommen, dass etwas nicht so funktioniert, wie Sie es sich vorgestellt haben. Sie werden manchmal Stunden damit verbringen den Quellcode wieder und wieder durchzuforschen, auf der Suche nach dem Fehler - um am Ende feststellen zu müssen, dass es wieder einmal ein kleiner Tippfehler war. Lassen Sie sich davon nicht entmutigen, denn auch so etwas gehört zum Lernen dazu.

Geübte Leser werden dieses Tutorial an einem Tag durchgelesen haben. Ich gehe aber davon aus, dass Sie erst nach einem Monat soweit sind, eigene Programme in BASIC zu schreiben. Ich verspreche Ihnen aber, die Zeit bis dahin wird spannend, interessant und abwechslungsreich werden.

BASIC-Programmierung ist der ideale Einstieg in die Welt der Programmierung. Gerade wenn Sie noch gar keine Erfahrung mit irgendeiner Programmiersprache haben, kann ich Ihnen BASIC wärmstens empfehlen.

Aber warum gerade BASIC, zu einer Zeit, in der Sprachen wie Visual C++, C#.NET, Delphi oder Java gibt (um nur einige zu nennen). Ist BASIC nicht längst veraltet? Macht es nicht mehr Sinn gleich in eine aktuelle Hochsprache einzusteigen, und so was Antiquarisches wie dieses BASIC einfach links liegen zu lassen?

Nein, denn gerade beim Erlernen einer Programmiersprache finde ich es sehr wichtig, wenn man sich dabei auf das Wesentliche konzentrieren kann. Und bei BASIC ist das der Fall, denn Sie müssen sich nicht um andere Dinge kümmern. Später werden Sie es mit ihrem Wissen viel leichter haben, auch andere Programmiersprachen zu lernen.

In diesem Sinne, wünsche ich Ihnen mit den folgenden Kapiteln viel Spaß!

- Grundlagen
- Mit Basic rechnen
- Variable speichern Zahlen und Texte
- Tipps für das Programmieren
- Entscheidungen treffen
- Anweisungen wiederholen
- Beispielprogramme
- Anhang

## 4 Grundlagen

In diesem Kapitel werden die Grundlagen des Programmierens vorgestellt.

### 4.1 Was ist Programmieren?

Computer<sup>3</sup> können den schnellsten Prozessor, den größten Speicher oder die schönsten Zusatzgeräte haben – ohne ein Programm funktionieren sie nicht. Es spielt auch keine Rolle, ob es sich um einen kleinen Taschenrechner, um einen Desktop PC oder um eine gigantische Mainframe handelt – Software ist das, was sie so dringend benötigen, wie Fische das Wasser.

Diese Programme sind im Speicher des Computers abgelegt. Und weil der Speicher digital ist, wird die Information im Binären Format festgehalten. Es findet sich darin eine Folge von genau zwei verschiedenen Zeichen: 0 und 1. Diese kleinste Einheit, die ein Computer versteht wird auch Bit<sup>4</sup> genannt. Eine Gruppe von 8 Bits nennt man Byte.

Jeder Prozessor versteht eine gewisse Anzahl von Befehlen, die jeweils aus einer festgelegten Bitfolge bestehen. Diese Befehle, die ein Computer direkt verstehen und verarbeiten kann, werden Maschinensprache-Befehle genannt.

Eine Folge von Befehlen, die irgendetwas Nützliches zustande bringt, heißt Programm. Ist auch nur ein Bit in dieser Folge falsch, kann es der Computer nicht mehr verstehen, und er meldet einen Fehler - oder er stürzt sogar ab und muss neu gestartet werden. Der Computer führt ein Programm aus, indem er nacheinander jeden einzelnen Befehl ausführt.

Unglücklicherweise ist kaum jemand in der Lage ein Computerprogramm als Folge von Nullen und Einsen einzugeben. Es ist ein langsames und umständliches Verfahren. Deshalb wurden die Programmiersprachen entwickelt, die das Erstellen von Programmen erleichtern sollen. Eine Programmiersprache ist eine Sammlung von Regeln (der Syntax), Wörtern und Symbolen (dem Wortschatz), die es gestatten, dem Computer in einem ihm verständlichen Format Befehle zu erteilen.

Es gibt zwei verschiedene Arten von Programmiersprachen: Assemblersprachen und höhere Sprachen. Eine Assemblersprache ist eine symbolische Darstellung der Maschinensprache-Befehle. Das ist zwar wesentlich einfacher wie die Eingabe der entsprechende Bitfolge, bleibt aber immer noch schwierig in der Handhabung.

Höhere Programmiersprachen wurden geschaffen, um dieses Problem zu beseitigen. Basic<sup>5</sup> ist eine solche höhere Programmiersprache, und zudem noch diejenige, welche besonders für Programmier-Anfänger leicht erlernbar und daher bestens geeignet ist.

---

<sup>3</sup> Computer: Ihr Casio FX-850P/FX-880P kann getrost als solcher bezeichnet werden.

<sup>4</sup> Bit: Eine Zusammenfassung der Wörter **b**inary **d**igit (binäre Ziffer)

<sup>5</sup> BASIC: **B**eginners **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode (Symbolischer Allzweckcode für Anfänger)

Es gibt 2 Möglichkeiten, wie ein Programm zur Ausführung gelangen kann, das in einer höheren Programmiersprache geschrieben wurde: Entweder übersetzt ein *Interpreter* (siehe 12.2.1) zur Laufzeit Konstrukt für Konstrukt in Maschinencode und lässt diesen ausführen (genau das macht unser Casio Taschenrechner), oder aber, es findet eine zweistufige Behandlung des Programms statt. Zuerst wird der gesamte Code von einem *Compiler* (siehe 12.2.2) in Maschinencode übersetzt und abgespeichert. Danach kann der abgespeicherte Code so oft wie gewünscht, ohne neu übersetzt zu werden, einfach ausgeführt werden.

## 4.2 Was ist Basic?

Basic steht für **B**eginners **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode (Symbolischer Allzweckcode für Anfänger). Es wurde von 1964 von John Kemeny und Thomas Kurtz am Dartmouth College erfunden. Das Ziel der beiden war es, eine Sprache zu entwerfen, die so einfach ist, dass sie von Anfängern benutzt werden kann. Es gelang ihnen dieses Ziel zu erreichen. Bis heute ist BASIC eine der Programmiersprachen, die am einfachsten zu erlernen sind.

Die einfache Sprache benötigte nur 4K (4.096 Bytes) Speicher auf einem Computersystem. Dies machte es möglich, dass seit Ende der siebziger Jahre fast alle Mikrocomputer damit ausgestattet wurden. So z.B. Sinclair ZX81 Commodore C64, um nur die bekanntesten davon zu nennen. Aber auch bei den Herstellern von programmierbaren Taschenrechnern wurde BASIC gerne eingesetzt (so. z.B. bei unserem Casio aber auch bei SHARP- oder TI-Taschenrechnern).

Heute wird BASIC auf fast allen Computern benutzt. In all den Jahren haben die verschiedenen Hersteller Erweiterungen und neue Möglichkeiten der Sprache hinzugefügt, so dass heutzutage BASIC die Computersprache ist, die am geringsten standardisiert ist. Nicht einmal zwei BASIC-Versionen sind gleich. Tatsache ist, dass BASIC eine Familie von Sprachen geworden ist und nicht mehr nur eine einzige Sprache darstellt.

Obgleich viele Standards vorgeschlagen wurden, hat sich keiner durchgesetzt, und es ist sehr unwahrscheinlich, dass dies zu diesem späten Zeitpunkt noch geschieht. Heißt das nun, dass man für jeden Computer oder Taschenrechner BASIC neu lernen muss? Nicht ganz. Wenn Sie einmal mit den wesentlichen Begriffen von BASIC, die in allen Versionen vorkommen, vertraut sind, können Sie ganz leicht die Verbesserungen erlernen, die jede Version anbietet. Jedes BASIC hat im Wesentlichen den gleichen Kern von Befehlen.

In diesem Tutorial geht es speziell um das BASIC des Taschenrechners Casio FX-850P/FX-880P. Dieses Casio-BASIC wird mit kleinen Abwandlungen auch auf vielen anderen Casio Taschenrechner eingesetzt. Deshalb kann dieses Tutorial auch für folgende Casio Taschenrechner verwendet werden:

FX-700P, FX-702P, FX-710P, FX-720P, FX-730P, FX-740P, FX-750P, FX-770P, FX-790P, FX-791P, FX-795P, FX-801P, FX-802P, FX-820P, FX-840P, FX-841P, FX-860P, FX-860Pvc, FX-870P, FX-890P, PB-80, PB-100, PB-100F, PB-110, PB-120, PB-200, PB-220, PB-240, PB-300, PB-400, PB-500, PB-500F, PB-700, PB-770, PB-770J, SuperCollege II, VX-2

### 4.3 Die Syntax des Casio-BASIC

Eine Programmiersprache ist eine Sammlung von Regeln (der Syntax), Wörtern und Symbolen (dem Wortschatz), die es gestatten, dem Computer in einem ihm verständlichen Format Befehle zu erteilen.

Ein Casio-BASIC Programm besteht also aus einer Reihe von verschiedenen Elementen: Schlüsselwörter, Befehlen, Anweisungen, Funktionen und Variablen.

#### 4.3.1 Schlüsselwörter

Casio-BASIC Schlüsselwörter wie z.B. PRINT, GOTO oder RETURN haben eine bestimmte Bedeutung für den Casio-BASIC Interpreter. Casio-BASIC interpretiert diese Schlüsselwörter als Teil von Anweisungen oder Befehlen.

Schlüsselwörter nennt man auch *reservierte Wörter*. Sie können nicht als Variablennamen verwendet werden, denn in diesem Fall würde das System sie als Befehl interpretieren. Innerhalb von Variablennamen ist die Verwendung von Schlüsselwörtern jedoch möglich.

Schlüsselwörter werden zur effizienten Ausnutzung des Speicherplatzes im System als „Tokens“ (1- oder 2-Byte Zeichen) abgespeichert.

#### 4.3.2 Befehle

Befehle und Anweisungen sind ausführbare Instruktionen. Im Unterschied zu Anweisungen werden Befehle üblicherweise im Direktmodus, d.h. auf der Befehlsebene des Interpreters ausgeführt. Befehle steuern gewöhnlich die Programmpflege, wie z.B. das Bearbeiten, Laden und Sichern von Programmen.

#### 4.3.3 Anweisungen

Bei Anweisungen, wie z.B. ON ERROR...GOTO handelt es sich um eine Gruppe von Casio-BASIC Schlüsselwörtern, die gewöhnlich in Casio-BASIC Programmzeilen als Teil eines Programms verwendet werden. Beim Programmablauf werden die Anweisungen dann in der Reihenfolge, in der sie erscheinen, ausgeführt.

#### 4.3.4 Funktionen

Der Casio-BASIC Interpreter führt numerische Funktionen und Zeichenfolgen-Funktionen aus. Funktionen liefern einen Rückgabewert – im Gegensatz zu Anweisungen oder Befehlen.

##### Numerische Funktionen

Der Casio-BASIC Interpreter kann bestimmte mathematische (arithmetische oder algebraische) Berechnungen durchführen. Er kann z.B. den Sinus (SIN), Cosinus (COS) oder Tangens (TAN) eines Winkels x berechnen.

##### Zeichenfolgen-Funktionen

Zeichenfolgen-Funktionen arbeiten mit Zeichenfolgen. So liefert die Funktion HEX# beispielsweise eine Hexadezimal-Zeichenfolge für einen im Argument spezifizierten Dezimalwert.

### 4.3.5 Variablen

Variablen sind bestimmte Gruppen von alphanumerischen Zeichen, denen Werte zugewiesen werden. Wenn Variablen in einem Casio-BASIC Programm verwendet werden, übergeben sie bei der Ausführung bestimmte Werte.

So ergibt ERR zum Beispiel den letzten im Programm aufgetretenen Fehler an und ERL die Stelle, an der der Fehler auftrat. Variablen können auch vom Benutzer oder über das Programm definiert werden und / oder geändert werden.

Alle Casio-BASIC Befehle, Anweisungen Funktionen und Variablen werden einzeln in der CASIO-Bedienungsanleitung zum Nachschlagen beschrieben.

### 4.3.6 Zeilenformat

Jedes der Casio-BASIC Elemente kann zu Programmabschnitten zusammengefügt werden, die Anweisungen genannt werden. Diese Anweisungen ähneln Sätzen in englischer Sprache. Diese Anweisungen werden dann nach einem logischen Schema miteinander verknüpft und bilden so Programme.

Für Casio-BASIC Programmzeilen gilt das folgende Format:

- nnnnn Anweisung [Anweisungen]
- nnnnn ist eine Zeilennummer
- Anweisung ist eine Casio-BASIC Anweisung

Eine Casio-BASIC Programmzeile beginnt immer mit einer Zeilennummer und muss mindestens ein Zeichen enthalten. **Mehr als 255 Zeichen in einer Programmzeile sind nicht zulässig.** Die Zeilennummern geben die Reihenfolge an, in der die Programmzeilen im Speicher abgelegt werden, und werden auch als Bezugspunkte für Verzweigungen und Bearbeitungsaufgaben verwendet. Die Programmzeile wird durch Drücken der **EXE** Taste abgeschlossen.

Abhängig vom logischen Aufbau eines Programms können mehrere Anweisungen in einer Zeile stehen. In diesem Fall müssen die einzelnen Anweisungen durch einen Doppelpunkt (:) voneinander getrennt sein. Am Anfang jeder Programmzeile muss eine Zeilennummer stehen. **Als Zeilennummern können beliebige Ganzzahlen im Bereich von 1 bis 65635 verwendet werden.** Gewöhnlich werden als Zeilennummern 10, 20, 30 usw. gewählt, um später eventuell zusätzliche Zeilennummern leichter einfügen zu können.

Da der Computer die Anweisungen in numerischer Reihenfolge ausführt, brauchen die zusätzlich eingefügten Programmzeilen nicht in der entsprechenden Reihenfolge auf dem Display zu erscheinen. Wenn Sie beispielsweise die Programmzeile 35 nach der Programmzeile 40 eingegeben haben, führt der Computer dennoch nach der Programmzeile 30 erst Zeile 35 und dann Zeile 40 aus. Auf diese Weise brauchen Sie ein Programm nicht noch einmal einzugeben, wenn Sie eine Programmzeile vergessen haben.

Die Zeilenlänge auf dem Casio Display beträgt 32 Zeichen. Wenn eine Anweisung diese Zeilenlänge überschreitet, erfolgt automatisch eine Zeilenschaltung und der Text wird auf der nächsten Zeile fortgesetzt. Erst durch Betätigen der **EXE** Taste wird dem Computer mitgeteilt, dass eine Programmzeile beendet ist. Drücken Sie nicht die **EXE** Taste, wenn Sie sich bei der Eingabe dem Zeilenende nähern (oder darüber hinaus schreiben). Die Zeilenschaltung wird automatisch vom Computer vorgenommen.

Casio-BASIC interpretiert jede Textzeile, die mit einem numerischen Zeichen beginnt, als Programmzeile und verarbeitet sie nach Betätigen der **EXE** Taste in einer der drei folgenden Vorgehensweisen:

- **Eine neue Zeile wird an das Programm angefügt.**

Dies ist dann der Fall, wenn eine Zeilennummer gültig ist (im Bereich von 1 bis 65535 liegt) und mindestens ein Alphazeichen oder Sonderzeichen auf die Zeilennummer folgt.

- **Eine bestehende Zeile wird geändert.**

Dies ist dann der Fall, wenn die Zeilennummer mit einer im Programm existierenden Zeile übereinstimmt. Die vorhandene Zeile wird durch den Text der neu eingegebenen Zeile ersetzt. Dieser Vorgang wird *Bearbeiten* (Editieren) genannt.

**Hinweis**

Bei Verwendung einer bereits bestehenden Zeilennummer gehen alle Informationen in der Originalzeile verloren. Beachten Sie dies bitte bei der Eingabe von Zahlen im indirekten Modus. Sie könnten sonst versehentlich Programmzeilen löschen.

- **Eine vorhandene Zeile wird gelöscht.**

Dies ist dann der Fall, wenn die Zeilennummer mit einer im Programm existierenden Zeilennummer übereinstimmt und die eingegebene Zeile nur die Zeilennummer enthält. Beim Versuch, eine nicht vorhandene Zeile zu löschen, wird keine Fehlermeldung ausgegeben.

Wenn Sie Programme direkt auf dem Casio Rechner eintippen, sollten Sie sich folgende Manuelle Befehle genauer anschauen:

- EDIT           Damit schalten Sie den Edit-Modus
- LIST           Damit listen Sie ihr BASIC-Programm auf
- LIST ALL       Damit listen Sie ihr BASIC-Programm auf
- NEW            Damit löschen Sie das aktuelle BASIC-Programm
- NEW ALL        Damit löschen Sie alle BASIC-Programme
- RUN            Damit starten Sie ihr BASIC-Programm

## 4.4 Der Wortschatz des Casio-BASIC

Der Wortschatz des Casio-BASIC kann in folgende Kategorien aufgeteilt werden:

- Manuelle Befehle
- Grundbefehle
- Numerische Funktionen
- Zeichenfolge-Funktionen
- Eingabe/Ausgabe-Befehle
- Datenbank-Befehle

### 4.4.1 Manuelle Befehle

CLEAR, EDIT, FRE, LIST, LIST ALL, NEW, NEW ALL, PASS, RUN, TROFF, TRON, VARLIST

### 4.4.2 Grundbefehle

BEEP, CLS, DATA, DEFSEG, DIM, END, ERASE, ERL, ERR, FOR ~ NEXT, GOSUB, GOTO, IF ~ THEN ~ ELSE, IF ~ GOTO ~ ELSE, INKEY\$, INPUT, INPUT\$, LET, LOCATE, ON ERROR GOTO, ON GOSUB, ON GOTO, PEEK, POKE, PRINT, READ, REM ( ' ), RESTORE, RESUME, RETURN, SET, STOP, TAB

### 4.4.3 Numerische Funktionen

ABS, ACS, ANGLE, ASN, ATN, COS, CUR, EXP, FACT, FIX, FRAC, HYP ACS, HYP ASN, HYP ATN, HYP COS, HYP SIN, HYP TAN, INT, LN, LOG, NCR, NPR, PI, POL, RAN#, REC, ROUND, SGN, SIN, SQ R, TAN

### 4.4.4 Zeichenfolgen-Funktionen

&H, ASC, CHR\$, DEG, DMS\$, HEX\$, LEFT\$, LEN, MID\$, RIGHT\$, STR\$, VAL, VALF

### 4.4.5 Eingabe/Ausgabe-Befehle

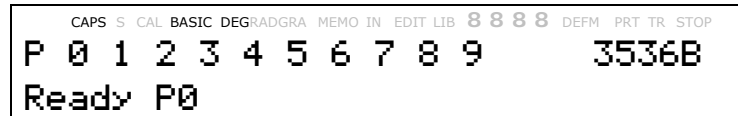
CLOSE, EOF, INPUT#, INPUT\$, LLIST, LOAD, LOAD ALL, LPRINT, OPEN, PRINT#, SAVE, SAVE ALL, VERIFY

### 4.4.6 Datenbank-Befehle

LIST#, LLIST#, LOAD#, NEW#, READ#, RESTORE#, SAVE#, WRITE#

## 4.5 Die Betriebsarten

Wenn beim Casio in den BASIC-Modus umgeschaltet wurde (dieser Modus wird durch das Drücken der Tasten **MODE** **1** aktiviert), wird auf dem Display „Ready P0“ angezeigt. „Ready P0“ bedeutet, dass der Casio Taschenrechner sich auf der Befehlsebene befindet, also bereit ist, Befehle entgegen zu nehmen.



Zu diesem Zeitpunkt kann der Casio nun in zwei verschiedenen Betriebsarten verwendet werden: im *Direktmodus* und im *indirekten Modus*.

### 4.5.1 Direktmodus

Im Direktmodus erfolgt die Anweisung der Basic Anweisungen und Befehle sofort bei der Eingabe. Die Ergebnisse arithmetischer und logischer Operationen können sofort angezeigt, oder für späteren Gebrauch gespeichert werden. Die Instruktionen selbst gehen jedoch bei der Ausführung verloren, d.h. sie werden nicht gespeichert. Diese Betriebsart eignet sich für die Fehlersuche und ermöglicht es, Rechenoperationen auszuführen, für die kein komplettes Programm erforderlich ist.

### 4.5.2 Indirekter Modus

Der indirekte Modus wird für die Eingabe von Programmen verwendet. Am Anfang der Programmzeilen steht immer eine Zeilennummer. Die Programmzeilen werden im Speicher abgelegt. Das im Speicher befindliche Programm wird durch Eingabe des Befehls RUN ausgeführt. Ein laufendes Programm kann mit der Taste **BRK** wieder unterbrochen werden.

## 4.6 Wie speichert bzw.- löscht man Programme?

Es gibt 10 Programmbereiche (P0 bis P9) in denen jeweils ein eigenes Basic-Programm abgelegt werden kann. Dieser Programmbereich kann im Basic-Modus über die Tastenfolge **S** gefolgt von **0** bis **9** gewechselt werden. Programme bleiben nach der Eingabe erhalten. Man muss also nicht Sichern, wie man das z.B. von einem Editor-Programm auf einem PC gewöhnt ist.

Nach dem Einschalten des Rechners stehen die eingegebenen Programme also sofort zur Verfügung.

Wenn ein Programm eines Programmbereichs gelöscht werden soll, verwendet man den Befehl NEW. Sollen alle Programme in allen Programmbereichen gelöscht werden, verwendet man NEW ALL. Einzelne Zeilen eines Programms werden durch die Eingabe der Zeilennummer, gefolgt von **EXE** gelöscht.

## 4.7 Wie überträgt man Programme vom bzw. zum PC?

Das Bearbeiten eines BASIC-Quelltextes auf dem PC oder Notebook ist um vieles einfacher und bequemer als direkt auf dem Casio Rechner. Außerdem sind so die BASIC-Listings sicher aufgehoben und können sogar ausgedruckt werden. Deshalb editiere ich selbst kleine Programme nicht mehr auf dem Casio, sondern auf dem PC und übertrage sie einfach nach der Fertigstellung.

Zur Datenübertragung benötigen Sie neben den Casio Taschenrechner einen PC, ein spezielles Editor-Programm (welches die Datenübertragung unterstützt) und ein Übertragungskabel.

Ein solches Editor-Programm ist z.B. **MB Casio Notepad**, welches Sie auf meiner Homepage herunterladen können. Mit MB Casio Notepad können Sie Memo-Daten oder Basic-Files von und zum Taschenrechner übertragen. Aber auch ein Übertragungskabel, ein **USB-Interface** speziell für den Casio Taschenrechner, biete ich auf meiner Homepage an.

### 4.7.1 Daten zum Casio senden

Im Programm-Menü wählen Sie bitte das Menü "Casio", und dort den Menüpunkt "Daten zum Casio senden...".

Sie können auch die Tastenkombination Strg+W dafür verwenden.

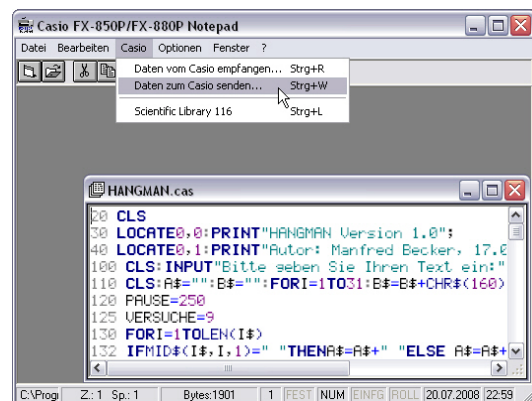


Abbildung 4

Es öffnet sich ein Transfer-Dialog, der Ihnen die folgenden Schritte erklärt.

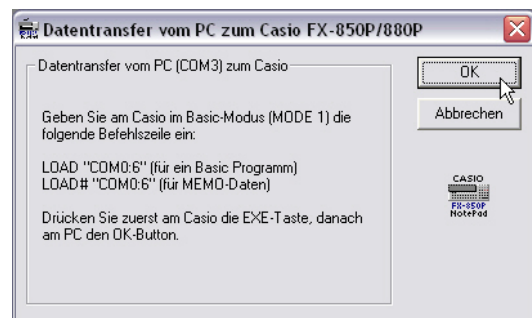


Abbildung 5

Während der Datenübertragung wird folgende Fortschritts-Anzeige eingeblendet.

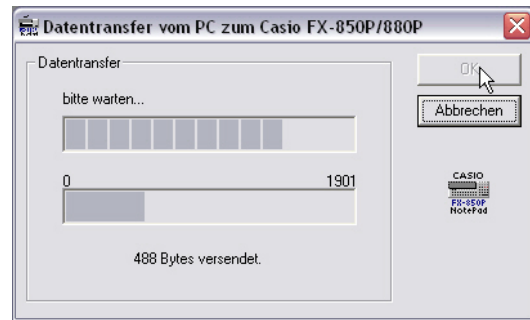


Abbildung 6

Nach der Datenübertragung erhalten Sie diese Statusmeldung.

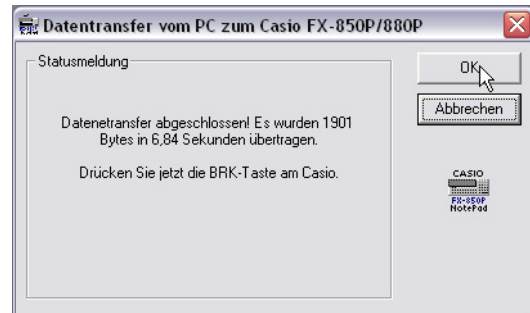


Abbildung 7

#### 4.7.2 Daten vom Casio empfangen

Im Programm-Menü wählen Sie bitte das Menü "Casio", und dort den Menüpunkt "Daten vom Casio empfangen...".

Sie können auch die Tastenkombination Strg+R dafür verwenden.

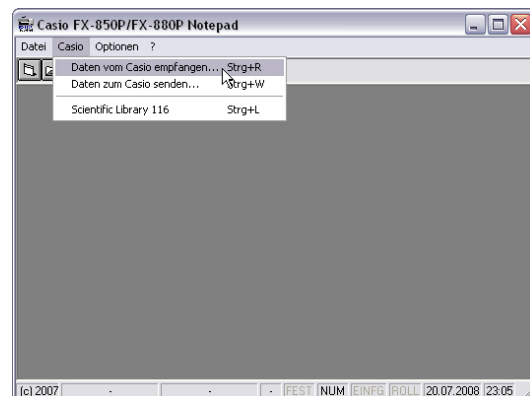


Abbildung 8

Es öffnet sich ein Transfer-Dialog, der Ihnen die folgenden Schritte erklärt.



Abbildung 9

Nach dem drücken der OK-Taste wartet das Programm auf die Daten vom Casio.

Jetzt haben Sie Zeit, um auf dem Casio den notwendigen Sende-Befehl einzugeben.



Abbildung 10

Während der Datenübertragung wird folgende Fortschritts-Anzeige eingeblendet.



Abbildung 11

Nach der Datenübertragung erhalten Sie diese Statusmeldung.

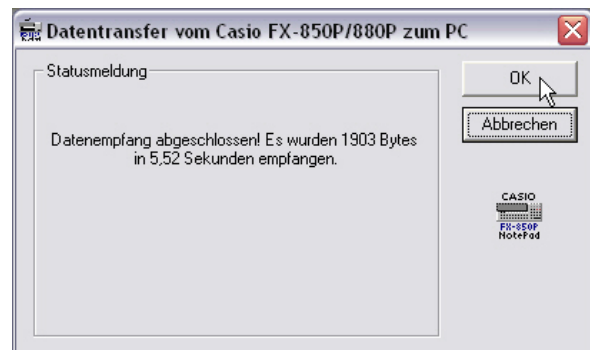


Abbildung 12

Die Datei kann nach der Datenübertragung editiert und gespeichert werden.

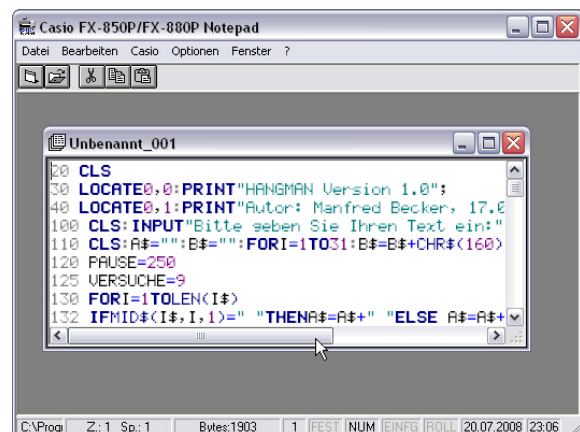


Abbildung 13

## 5 Mit Basic rechnen

In diesem Kapitel fangen wir an, mit Zahlen zu arbeiten, und auf dem Display auszugeben.

### 5.1 Die Ausgabe von Zahlen

Die Ausgabe von Zahlen (aber auch von Text) erfolgt mit dem Basic-Befehl **PRINT**. Die Ausgabe auf dem Display erfolgt von der gegenwärtigen Cursor-Position nach rechts. Wenn der Cursor die letzte Spalte der letzten Zeile auf dem Display erreicht, erfolgt ein Zeilenvorschub, wobei der gesamte Bildschirminhalt nach oben gescrollt wird. Dann erfolgt die nächste Ausgabe vom Anfang der untersten Zeile des Displays.

Zahlen werden ohne Anführungszeichen direkt nach der PRINT-Anweisung angegeben.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT 5
```

Die Ausgabe auf dem Display erfolgt in der nächsten Zeile.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
5
```

Nach der Anzeige von numerischen Ausdrücken wird ein Leerzeichen angefügt, negativen Ausdrücken wird ein Minuszeichen und positiven Ausdrücken wird eine Leerstelle vorangestellt.

### 5.2 Wissenschaftliche Schreibweise

Numerische Ausdrücke werden in der Dezimalnotation angezeigt. Dabei werden Werte, die länger sind als 10 Stellen, mit einer auf 10 Stellen gerundeten Mantisse und einem 2-stelligen Exponent angezeigt.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT 12345678900
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
1.23456789E+10
```

Die Eingabe kann aber genauso gut in dieser Exponentialdarstellung erfolgen.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT 1.23E-6
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
0.00000123
```

### 5.3 Mathematische Regeln

Sie können auch das Ergebnis einer arithmetischen Berechnung ausgeben lassen. Dabei wird das Ergebnis nach folgender Prioritätsfolge berechnet:

1. Klammern  $()$
2. Funktionen
3. Potenz
4. Vorzeichen  $+, -$
5.  $*, /, \%, \text{MOD}$
6.  $+, -$
7. Vergleichsoperatoren
8. NOT
9. AND
10. OR, XOR

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
PRINT 1+2*3
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
7
```

Um die Reihenfolge der Operationen zu beeinflussen, können Sie Klammern verwenden.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
PRINT (1+2+3)/(4+5)
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
0.6666666667
```

Folgende arithmetische Operatoren werden vom Casio-BASIC unterstützt:

- Subtraktion
- + Addition
- \* Multiplikation
- / Division
- ^ Potenzieren
- % Ganzzahlige Division
- MOD Ganzzahliger Rest von ganzzahliger Division

## 5.4 Formatierte Ausgabe

Bei der Ausgabe von Daten durch die PRINT-Anweisung können Sie mehrere Ausgabedaten angeben, die jeweils durch Kommata oder Semikolons getrennt sind.

Wenn die Daten durch Kommata getrennt werden, wird die Ausführung bei jeder Anzeige unterbrochen (STOP erscheint auf dem Display). Durch Drücken von **EXE** wird ein Wagenrücklauf/Zeilenvorschub ausgeführt, und zur nächsten Anzeige weitergegangen.

Wenn die Daten durch Semikolons getrennt werden, wird jede Ausgabe unmittelbar hinter der vorhergehenden Ausgabe angezeigt. Wird am Ende einer PRINT-Anweisung ein Semikolon gesetzt, bleibt der Cursor in der Position unmittelbar hinter der angezeigten Ausgabe. Außerdem erfolgt dann keine Unterbrechung bei der Programmausführung.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP  
PRINT "1 kByte=";2^10;"Bytes"
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP  
1 kByte= 1024 Bytes
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP  
PRINT 1;2;3
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP  
1 2 3
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP  
PRINT "A";"B";"C"
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP  
ABC
```

## 5.5 Übungen

### 5.5.1 Übung 5-1

Schreiben Sie einen BASIC-Befehl, der folgendes berechnet:

$$\frac{5 + 6}{1 + 2 : 3}$$

### 5.5.2 Übung 5-2

Schreiben Sie einen BASIC-Befehl, der folgendes berechnet:

$$1+1/2 \frac{1}{1 + 1/2}$$

### 5.5.3 Übung 5-3

Schreiben Sie einen BASIC-Befehl, der die Temperatur von 20 Grad Celsius in den entsprechenden Fahrenheit-Wert umrechnet, mit Hilfe der Formel:

$$\text{CELCIUS-WERT} = (\text{FAHRENHEIT-WERT} - 32) * 5 / 9$$

### 5.5.4 Übung 5-4

Die Erde dreht sich pro Tag einmal um ihre eigene Achse. Mit welcher Geschwindigkeit bewegt sich dann ein Punkt des Äquators? (Der Erdumfang beträgt ca. 40.000 km)

### 5.5.5 Übung 5-5

Berechnen Sie die Sekunden eines Tages, einer Woche, eines Monats und eines Jahres.

### 5.5.6 Übung 5-6

Gegeben ist eine Fahrgeschwindigkeit von 55 km/h. Berechnen Sie, wie lange Sie für 350km benötigen.

### 5.5.7 Übung 5-7

In einem rechtwinkligen Dreieck gilt:  $c^2 = a^2 + b^2$  (Satz des Pythagoras; a,b: Katheten, c: Hypotenuse). Berechnen Sie c, wenn a=3 und b=4 ist. (Hinweis: Die Wurzelfunktion heißt SQR)

### 5.5.8 Übung 5-8

In einem rechtwinkligen Dreieck gilt:  $h^2 = p * q$  (Höhensatz; h: Höhe, p,q: Hypotenusenabschnitte). Außerdem gilt:  $a^2 = c * p$ ;  $b^2 = c * q$  (Kathetensatz nach Euklid). Berechne die Höhe h des rechtwinkligen Dreiecks aus Übung 7.

### 5.5.9 Übung 5-9

Schreiben Sie einen BASIC-Befehl mit den Variablen a=1 und b=2, der folgende Ausgabe bewirkt:

Die Summe von  $1 + 2 = 3$

## 6 Variable speichern Zahlen oder Texte

In diesem Kapitel lernen Sie den Umgang mit Variablen. Außerdem erfahren Sie, wie man während des Programmablaufes Daten über die Tastatur eingeben kann.

### 6.1 Zwei Arten von Variablen

Variablen sind Namen, die Sie zur Darstellung von Werten verwenden können. Der Wert einer Variablen kann explizit von Ihnen zugewiesen werden, oder als Benutzereingabe erfolgen. Außerdem kann das Ergebnis von Berechnungen im Programm den Wert einer Variablen bestimmen.

Nach der Wertezuweisung merkt sich der Rechner den Inhalt jeder Variable, und kann so an einer anderen Stelle wieder beliebig abgerufen werden. Erst beim Zurücksetzen des Rechners (RESET oder ALL RESET), oder beim Löschen des Variablenspeichers mit dem Basic Befehl **CLEAR** bzw. **NEW** oder **NEW ALL**, geht der Variableninhalt verloren. Wird einer Variablen kein Wert zugewiesen, geht das CASIO-Basic davon aus, dass der Wert der Variablen Null ist.

Variablennamen können beliebig lang sein; für den CASIO-Basic Interpreter sind jedoch nur maximal 15 Zeichen von Bedeutung. Variablennamen können Buchstaben und Zahlen enthalten. Das erste Zeichen muss jedoch immer ein Buchstabe sein. Reservierte Wörter (Alle CASIO-Basic Befehle, Anweisungen, Funktionen und Operatoren) können nicht als Variablenname verwendet werden. Im Variablennamen eingeschlossene reservierte Wörter sind jedoch zulässig.



#### Achtung

CASIO-Basic unterscheidet beim Variablennamen zwischen Groß- und Kleinschreibung! Das ist bei vielen BASIC-Dialekten nicht üblich. Denken Sie bitte immer daran!

Variablen stellen entweder **numerische Werte** oder **Zeichenfolgen** dar. Die Unterscheidung wird bei Zeichenfolgen-Variable durch das Anhängen des String-Zeichens **\$** an den Variablennamen getroffen.



#### Achtung

Es ist beim CASIO-Basic zulässig, eine namensgleiche Numerische- und Zeichenfolge-Variable gleichzeitig zu verwenden! Ich empfehle besser nicht davon Gebrauch zu machen, um Verwechslungen zu vermeiden!

Variablen haben sowohl im CAL-Modus als auch im BASIC-Modus ihre Gültigkeit. So kann man beispielweise bei Berechnungen, Zwischenergebnisse in Variablen abspeichern.

### 6.1.1 Numerische Variable

Numerische Variable nehmen Zahlenwerte auf. Wobei es ganz egal ist, ob es eine Ganzzahl (Integerwert), oder eine Kommazahl (Floatwert) ist. Das CASIO-Basic macht keinerlei Unterscheidungen zwischen den verschiedenen Numerischen Zahlen-Typen.

Hier wird der Variablen R der Wert 5 zugewiesen. Dann wird die Fläche eines Kreises berechnet und der Variablen A zugewiesen.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
R=5
A=PI*R^2
```

Bei dieser Berechnung kommt auch unsere Variable R vor. Den Inhalt von Variablen kann man mit dem Befehl PRINT ausgeben lassen.

Hier sehen Sie den Inhalt von R, und das Rechenergebnis, welches in A gespeichert ist.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT R:A
5 78.53981634
```

Wie bereits erwähnt bleibt der Variableninhalt auch nach dem Ausschalten erhalten! Wollen Sie Variableninhalte explizit löschen, können Sie den Befehl **CLEAR** verwenden.

### 6.1.2 Zeichenfolge-Variable

Zeichenfolgen (oder String-Variable) können ebenfalls im CAL-Modus verwendet werden. Allerdings muss bei der Zuweisung die Zeichenfolge selbst in Hochkommas eingeschlossen werden. Und dem Variablennamen muss mit dem \$-Zeichen enden.

Hier werden zwei Variable mit einer Zeichenfolge belegt.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
N$="Mein Casio ist super!"
M$="4ever"
```

Auch hier erfolgt die Ausgabe des Variableninhalts mit dem PRINT-Befehl.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT N$
Mein Casio ist super!
```

Auch die Zeichen-Funktionen lassen sich auf Variable anwenden.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT MID(N$,6,5);M$
Casio4ever
```



#### Hinweis

Variable werden später beim Programmieren noch sehr hilfreich sein.

Hier noch ein paar Beispiele für Variablennamen:



### Beispiel

Variablennamen	Bemerkung
ABC	Gültige numerische Variable.
ASDF	Ungültige numerische Variable, weil sie mit dem reservierten Wort „AS“ beginnt.
REPRINT	Gültige numerische Variable, denn das reservierten Wort "PRINT" befindet sich nicht am Anfang
Name123\$	Gültige Zeichenfolgen-Variable
AllesIstOk\$	Ungültige Zeichenfolgen-Variable, weil sie mit dem reservierten Wort „ALL“ beginnt.
1UND1\$	Ungültige Zeichenfolgen-Variable, weil sie mit einer Ziffer beginnt.
A_B	Ungültige numerische Variable. Der Unterstriche ist, wie alle anderen Sonderzeichen, nicht erlaubt.

Wenn Sie bei einem BASIC-Programm nicht genau wissen, welchen Namen Sie Ihren Variablen geben sollen, dann rate ich Ihnen, aussagekräftige Namen zu verwenden. Dadurch ist das BASIC-Programm leichter zu verstehen. Vor allem müssen Sie nicht lange rätseln, wenn Sie nach Wochen oder Monaten Ihr BASIC-Programm erweitern wollen, und dann nicht mehr wissen, wozu genau die einzelnen Variable notwendig waren.

Da Variable auch Speicherplatz benötigen, sollten Sie noch einen weiteren Rat im Umgang mit Variable beherzigen: Verwenden Sie so wenig wie möglich, aber so viele wie nötig.



### Hinweis

Eine Übersicht aller Variablen erhalten Sie übrigens mit dem Befehl **VARLIST**.

## 6.2 Der INPUT-Befehl

Mit dem INPUT-Befehl weisen Sie Daten, die über die Tastatur eingegeben wurden Variablen zu. Damit ist es möglich den Anwender aufzufordern, Eingabedaten anzugeben. Interessanterweise verschweigt das Handbuch, dass man die Anzahl Zeichen auch hier begrenzen kann z.B. mit `INPUT@10;A`. Der INPUT-Befehl funktioniert nicht im CAL-Modus, sondern muss in einer Programmzeile eingebunden werden.

Weitere Tastatur-Eingabebefehle sind `INPUT$` und `INKEY$`, welche aber erst im nächsten Kapitel erklärt werden.

### 6.2.1 Eingabe von numerischen Variablen

Hier ein Beispielprogramm, bei dem der Anwender eine Zahleingabe muss.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 INPUT A
20 PRINT "Ihre Eingabe war";A
```

Der Programmstart erfolgt mit dem BASIC-Befehl RUN.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
RUN
?
```

Die Benutzereingabe muss mit der **EXE** Taste abgeschlossen werden.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
?123
Ihre Eingabe war 123
```

### 6.2.2 Eingabe von Zeichenfolgen-Variablen

Hier ein Beispielprogramm, bei dem der Anwender eine Zeichenfolge eingeben muss.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 INPUT "Ihr Name";NAME$
20 PRINT "Hallo ";NAME$;"!"
```

Der Programmstart erfolgt mit dem BASIC-Befehl RUN.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
RUN
Ihr Name?
```

Auch hierbei mit der **EXE** Taste abschließen.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
Ihr Name?Manfred
Hallo Manfred!
```

### 6.2.3 Verwendung mehrerer Variablen bei einem INPUT-Befehl

Bei einem INPUT-Befehl dürfen Sie mehrere, durch Komma getrennte, Variable angeben. Beim Programmablauf werden dann nacheinander die einzelnen Variablen abgefragt.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 INPUT A,B,C
20 INPUT "1";A$,"2";B$,"3";C$
```

### 6.3 Feldvariable

Sowohl numerische Variable als auch Zeichenfolge-Variable können jeweils nur einen einzigen Wert speichern. Will man aber z.B. für die Berechnung einer Wertetabelle viele Einzelwerte speichern, greift man besser auf Feldvariable zurück.

Ein Feld (array) ist eine Gruppe oder Tabelle von Werten, die über den gleichen Variablennamen angesprochen werden. Die einzelnen Elemente in einem Feld werden über einen Index, welcher in Klammern eingeschlossen ist, angesprochen.

Bevor eine Feldvariable in einem Programm verwendet werden kann, muss sie vereinbart (deklariert) werden. Die Deklaration erfolgt mit der DIM-Anweisung.

#### Beispiel:

Wenn Sie z.B. die Wertetabelle zu der Funktion:  $f(x) = 0.5x^3 - 2x^2 + 3x - 4$  Berechnen und speichern wollen, wobei  $x$  die Werte von 0 bis 20 annehmen soll, dann verwendet Sie hierzu nicht 21 Variable, sondern eine einzige Feldvariable:

In **Zeile 10** wird die Feldvariable  $f(20)$  deklariert. Die Angabe 20 bei der Deklaration bedeutet, dass Sie Indexwerte von 0 bis 20 verwenden können. In der **Zeile 20 und 50** wird eine FOR-NEXT Schleife<sup>6</sup> verwendet, um die Variable  $x$ , die später als Indexwert für die Feldvariable dienen soll, nacheinander mit den Werten 0, 1, 2, ...bis 20 zu belegen. In der **Zeile 30** wird der Feldvariablen  $f(x)$  der jeweilige Funktionswert zugewiesen. Und in der **Zeile 40** wird der Indexwert  $x$  und der Funktionswert, welcher in der Feldvariablen  $f(x)$  gespeichert ist, ausgegeben.

```

CAPS S CAL BASIC DEG/GRD/GR A MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
10 DIM f(20)
20 FOR x=0 TO 20
30  f(x)=.5*x^3-2*x^2+3*x-4
40  PRINT x; f(x)
50 NEXT x

```

#### 6.3.1 Eindimensionale Felder

Im obigen Beispiel wurde ein eindimensionales Feld verwendet, weil es aus einer einzigen Reihe von Elementen besteht. Felder können aber mehr als eine Dimension aufweisen. In diesem Fall spricht man auch von Mehrdimensionalen Feldern.

#### 6.3.2 Mehrdimensionale Felder

Mehrdimensionale Felder werden bei der DIM-Anweisung mit zwei oder mehr Index-Werten deklariert. So erhält man z.B. mit `DIM T(5,10)` ein zweidimensionales Feld, und mit `DIM C(2,2,2)` ein dreidimensionales Feld.



#### Hinweis

Soll ein Feld, welches bereits deklariert ist, umdeklariert werden, muss es zuvor mit `CLEAR` oder `ERASE` gelöscht werden.

<sup>6</sup> Die For – Next Schleife wird im Kapitel 9 erklärt.

## 6.4 Konstante

Konstante sind Variable, welche im gesamten Programmablauf ihren Wert nicht verändern. Das CASIO-Basic hat nur eine einzige Konstante, nämlich **PI**. Der Wert von PI ist bei Ihrem CASIO 3.1415926536. Sie können selbst keine eigenen Konstanten definieren. Verenden Sie anstelle einer Konstanten eine normale Variable, welche Sie beim Programmstart einmalig zuweisen, und während des Programms nicht mehr abändern.



**Hinweis**

Der Versuch PI einen anderen Wert zuzuweisen, wird mit einem SN error quittiert.

## 6.5 Wertzuweisungen

Der Wert einer Variablen kann von Ihnen explizit zugewiesen werden, oder als Ergebnis einer Berechnung im Programm entstehen. Zur Zuweisung muss der Befehl **LET Variable = Ausdruck** verwendet werden, wobei LET auch ausgelassen werden kann und die verkürzte Schreibweise **Variable = Ausdruck** angewendet werden kann.

Diese Zuweisung kann mit beliebig vielen Variablen erfolgen. Eine Deklaration (Bekanntmachung) des Variablennamens ist nicht erforderlich. So kann auch mitten im Programm auf eine Variable zugegriffen werden, die zuvor noch nicht verwendet wurde.

Das vereinfacht die Handhabung mit Variablen sehr, birgt aber auch die Gefahr, dass sich Programmfehler durch falsch geschriebene Variablennamen einschleichen können.

Finden Sie den Programmfehler dieses Zweizeilers auf Anhieb?

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 NAME$="Manfred"
20 PRINT "Hallo ";Name$;"!"
```

Nun, der Fehler liegt an der unterschiedlichen Schreibweise des Variablennamens. NAME\$ in Zeile 10 und Name\$ in Zeile 20 sind zwei unterschiedliche Variablen! Die in Zeile 20 gewünschte Ausgabe des in Zeile 10 zugewiesenen Namens kann so nicht funktionieren.

## 6.6 Variable verändern

Nun, wie bereits der Name Variable suggeriert, kann ihr Wert beliebig oft verändert bzw. neu zugewiesen werden. Der vorhergehende Wert wird dabei einfach überschrieben und ist dadurch verloren gegangen. Man kann den alten Wert nicht wieder zurückholen.

Die Ausgabe von diesem kleinen Programm ist, wie zu erwarten, zuerst „Manfred“ und nach Drücken der **EXE** Taste wird „Casio“ angezeigt.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 NAME$="Manfred"
20 PRINT "Hallo ";Name$;"!"
30 NAME$="Casio"
40 PRINT "Hallo ";Name$;"!"
```

## 6.7 Übungen

### 6.7.1 Übung 6-1

Schreiben Sie ein Programm, welches vier Zahlen von der Tastatur einliest und danach die Summe, den Mittelwert und das Produkt der vier Zahlen ausgibt.

### 6.7.2 Übung 6-2

Sind folgende Variablennamen richtig?

- |            |              |            |
|------------|--------------|------------|
| a.) 24B    | e.) ALPHA2D  | i.) PL     |
| b.) B24    | f.) BEISPIEL | j.) 3\$    |
| c.) A+B    | g.) INPUT    | k.) DREI   |
| d.) APLUSB | h.) INPUT1   | l.) NAME\$ |

### 6.7.3 Übung 6-3

Schreiben Sie ein Programm, das nach dem Namen des Anwenders fragt und danach folgendes ausgibt:

„Hallo (hier den Namen)! Ihr Name hat (hier Anzahl Buchstaben) Buchstaben.“

Hinweis: Die BASIC-Funktion, welche die Anzahl der Zeichen einer Zeichenfolge liefert heißt LEN()

### 6.7.4 Übung 6-4

Schreiben Sie ein Programm, das drei Seiten eines rechtwinkligen Dreiecks einliest (a,b und c). Dabei sind nur bei zwei beliebigen Seiten ein Wert > 0 einzugeben. Bei der gesuchten dritten Seite soll 0 eingegeben werden. Das Programm soll die gesuchte Seite berechnen und ausgeben.

### 6.7.5 Übung 6-5

Schreiben Sie ein Programm, das den Radius (r) eines Kreises einliest, und dann den Durchmesser (d), den Umfang (U) und die Fläche (A) berechnet und ausgibt.

### 6.7.6 Übung 6-6

Sind folgende Zuweisungen richtig?

- |               |                           |                     |
|---------------|---------------------------|---------------------|
| a.) $A+1=2$   | e.) $3=2+1$               | i.) $A\$=A\$+1$     |
| b.) $A=A+A+A$ | f.) $ZAHL=ERSTE+LETZTE*2$ | j.) $AB\$=A+B\$$    |
| c.) $A=B+C$   | g.) $A\$=A\$+B\$$         | k.) $"Test"=C\$$    |
| d.) $B+C=A$   | h.) $A\$=B\$-A\$$         | l.) $FORD\$="AUTO"$ |

### 6.7.7 Übung 6-7

Schreiben Sie ein Programm, das die Seitenlänge (a) eines Würfels einliest, und dann das Volumen (V), die Oberfläche (O) und die Raumdiagonale (d) berechnet und ausgibt.

## 7 Tipps für das Programmieren (Teil 1)

So, inzwischen sind Sie in der Lage einfachste Programme zu schreiben, welche Daten ausgeben bzw. Eingaben vom Anwender erwarten. Zugegeben das ist noch nicht wirklich berauschend, aber schon jetzt gibt es ein paar Tipps, die Sie beherzigen sollten.

### 7.1 Der REM-Befehl

Ein BASIC-Programm besteht aus einer Aneinanderkettung von Programmzeilen, welche wiederum irgendwelche BASIC-Anweisungen beinhalten. Mit der **REM**-Anweisung können Sie Kommentare einfügen, um Ihr Programm verständlicher zu machen. Schließlich wollen Sie ja auch noch nach Monaten oder Jahren wissen, wozu das Programm geschrieben wurde, und was es macht. Und das, ohne jede einzelne BASIC-Zeile zu untersuchen.

Das gezeigte BASIC-Listing ist mit den Kommentagen auch für andere BASIC-Programmierer viel leichter zu verstehen.

Ja sogar ein Laie kann mit den Kommentaren errahnen, wozu das BASIC-Programm geschrieben wurde.

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 REM *****
11 REM Addition zweier Zahlen
12 REM Autor: Manfred Becker
13 REM Datum: 29.07.2009
14 REM *****
20 REM Eingabe
30 INPUT "Summand 1";S1
40 INPUT "Summand 2";S2
50 REM Addition
60 S=S1+S2
70 REM Ausgabe des Ergebnisses
80 PRINT "Die Summe ist";S

```

REM-Anweisungen werden vom Interpreter ignoriert, wenn das Programm mit RUN ausgeführt wird. Sie haben keinerlei Auswirkungen auf das Programm. Anstelle von REM kann auch das einfache Hochkomma (') verwendet werden.

### 7.2 Mehrere Anweisungen in einer Zeile

Beim CASIO-Basic können (wie bei vielen anderen BASIC-Versionen auch) zwei oder mehrere Anweisungen durch einen Doppelpunkt getrennt in die gleiche Programmzeile geschrieben werden.

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 R=5
20 A=PI*R^2
30 PRINT R;A

```

Die drei Programmzeilen können auch zu einer zusammengefasst werden.

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 R=5:A=PI*R^2:PRINT R;A

```

Insgesamt darf eine Zeile aber nicht mehr wie 255 Zeichen enthalten!

Es gibt zwei Fälle in denen das Zusammenfassen in eine Programmzeile von Vorteil sein kann:

1. Um eine INPUT-Anweisung klarer zu machen:

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINT "Radius": INPUT R
```

2. Um REM-Anweisungen an die Programmzeile anzufügen:

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
20 A=PI*R^2: REM Kreisflaeche
```

### 7.3 Gebrauch von Leerzeichen

Mit Ausnahme von Namen, Zeichenfolgen und Eingabedaten werden Leerzeichen im Allgemeinen vom CASIO-Basic ignoriert.

Durch die sinnvolle Verwendung von Leerzeichen kann z-B. diese Programmzeile:

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINTA+6*B
```

sehr viel übersichtlicher und lesbarer dargestellt werden:

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINT A + 6 * B
```

Auch bei For-Next-Schleifen kann man mittels Leerzeichen die eingeschlossenen Programmzeilen (Zeile 30 und 40) etwas einrücken, um deutlich zu machen, dass diese zur For-Next-Schleife gehören.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 DIM f(20)
20 FOR x=0 TO 20
30   f(x)=.5*x^3 - 2*x^2 + 3*x -4
40   PRINT x; f(x)
50 NEXT x
```

### 7.4 Auswahl von Variablennamen

Bei Variablennamen sollten Sie diese so auswählen, dass Sie immer genau wissen was der Name darstellt. Das wird umso wichtiger, je größer Ihr Programm wird. Bei einem Dreizeiler brauchen Sie sich keine Gedanken zu machen, aber wenn das BASIC-Listing einmal mehrere Seiten lang wird, können gut gewählte Variablennamen die Lesbarkeit und die Verständlichkeit sehr erhöhen.

Es ist eine gute Idee, die Bedeutung aller Variablen zu Beginn eines jeden Programms als Kommentare anzugeben.

## 7.5 Richtige Zeilennummerierung

Bisher haben Sie gelernt, dass die Zeilennummern welche vor jeder Programmzeile geschrieben steht, im Bereich von 1 bis 65535 sein dürfen. In den einfachen Programmlistings wurde meist mit den Zeilennummern 10, 20, 30 usw. gearbeitet. Vielleicht haben Sie sich gefragt, warum man nicht bei der Zeilennummer 1 beginnt, und dann einfach immer mit der nächst größeren Zeilennummer weitermacht (also 1, 2, 3 usw.).

Die Antwort hängt mit der typischen Vorgehensweise beim Programmieren zusammen: Ein Programm wird nicht dadurch erstellt, dass man mit der ersten Zeile anfängt, und dann nacheinander alle weiteren Zeilen programmiert, bis man am Ende die Letzte Zeile geschrieben hat. Nein, ein Programm entsteht durch mehrere Entwürfe, wobei Teile davon korrigiert oder verbessert werden. Manche Teile werden sogar gelöscht, andere hinzugefügt.

Aus diesem Grund, um flexibel zu bleiben, sollte man immer genügend Spielraum zwischen zwei Zeilennummern haben. Es könnte ja sein, dass man ein oder zwei Zeilen dazwischen einfügen will (oder muss).

Deswegen sollten Sie Ihre Zeilennummern auch mit einem Vielfachen von 10 nummerieren. Die erste Zeilennummer sollte bei größeren Programmen mindestens 100 sein. Wenn das Programm Unterroutinen (Subroutines) oder DATA-Anweisungen besitzt, sollten diese ebenfalls weit abgesetzt vom Hauptprogramm nummeriert werden (z.B. ab 10000).

Im Gegensatz zu vielen anderen BASIC-Versionen besitzt das CASIO-Basic keine RENUMBER-Funktion, um im Nachhinein alle Zeilen automatisch neu durchnummerieren. Deshalb achten Sie bitte schon beim Beginn auf eine richtige Zeilennummerierung, denn eine nachträgliche Umnummerierung von Hand ist sehr aufwändig und fehleranfällig.

## 7.6 Bildschirmlöschen und positionsgenaue Ausgabe

Die Ausgabe auf dem Display erfolgt mit dem PRINT-Befehl, und zwar dort, wo der Cursor positioniert ist. Manchmal, oft zu Beginn eines Programms, möchte man den Bildschirm komplett löschen. Und es gibt ab und zu die Notwendigkeit eine Ausgabe immer an derselben Bildschirmposition machen zu können. Nun, für diese beiden Anforderungen hält das CASIO-Basic einen entsprechenden Befehl parat:

### 1. CLS

CLS löscht den Bildschirm, und der Cursor wird zur Ursprungsposition (obere linke Ecke) bewegt.

### 2. LOCATE

LOCATE x,y bewegt den Cursor zu einer angegebenen Position auf dem Bildschirm. Die Parameter x und y bestimmen die Koordinaten:

- X-Koordinate von 0 bis 32
- Y-Koordinate von 0 bis 8

Hier ein Auszug aus einem BASIC-Programm:

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
20 CLS
30 LOCATE 0,0: PRINT "HANGMAN
Version 1.0";
40 LOCATE 0,1: PRINT "Autor:
Manfred Becker, 17.05.94"
```

## 7.7 Weitere Eingabebefehle

Wie man Benutzereingaben realisiert haben Sie ja bereits gelernt. Sie können den INPUT-Befehl dazu verwenden. Beim INPUT-Befehl werden die eingetippten Zeichen auf dem Display angezeigt. Manchmal benötigt man Benutzereingaben ohne dass die Eingabe selbst zu sehen ist. Dafür kann man die Tastatur-Eingabefunktionen **INPUT\$** oder **INKEY\$** verwenden.

### 7.7.1 Die INPUT\$ Funktion

Typischerweise dürfen bei einer Passwordeingabe auf keinen Fall die eingegebenen Zeichen angezeigt werden, denn sonst bliebe das Passwort nicht sehr lange geheim. In einem solchen Fall kann zur Benutzereingabe die Funktion **INPUT\$** verwendet werden.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINT "Passwort eingeben";
20 PW$ = INPUT$(4)
```

... die hier folgenden Programmzeilen finden Sie in Kapitel 9

Auch bei einem Auswahlmenü ist eine Eingabe notwendig.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINT "1, 2 oder 3?";
20 z$=INPUT$(1)
```

... die hier folgenden Programmzeilen finden Sie in Kapitel 9

### 7.7.2 Die INKEY\$ Funktion

Manchmal muss man auf einen einzelnen Tastendruck reagieren können. Das erledigt die **INKEY\$**-Funktion. Sie liefert nämlich eine einzelne Zeicheneingabe von der Tastatur. Wenn keine Taste gedrückt wurde, dann wird eine Null-Zeichenfolge zurückgeliefert.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
110 PRINT "Nochmal? (j/n)";
120 CH$ = INKEY$
```

... die hier folgenden Programmzeilen finden Sie in Kapitel 9

## 8 Entscheidungen treffen

Es wird langsam Zeit etwas Aufregenderes wie Ein- oder Ausgaben zu programmieren. Eine Möglichkeit, die uns das CASIO-Basic bietet, ist es Entscheidungen zu treffen. Und diese Entscheidungen können entweder Aufgrund ganz einfacher Vergleiche von Werten, oder durch die kompliziertesten Logischen Verknüpfungen getroffen werden.

### 8.1 Die GOTO-Anweisung

Zunächst aber komme ich zur GOTO-Anweisung, welche gerade im Zusammengang mit der IF-Anweisung sehr häufig verwendet wird. Sie wissen bereits, dass der BASIC-Interpreter nacheinander alle Programmzeilen abarbeitet. Dabei gibt die Zeilennummer vor, in welcher Reihenfolge das passiert.

Die GOTO-Anweisung hingegen, veranlasst den Interpreter mit der Programmzeile weiterzumachen, welche explizit angegeben wird. Damit kann man gezielt zu einer Stelle des Programms springen, an der es weitergehen soll.



**Hinweis**

Sie können sogar die Bibliothek-Funktionen aus Ihrem Programm aufrufen, wenn Sie **GOTO "LIB0:NNNN"** verwenden, wobei NNNN die vierstellige LIB-Nummer ist.

Erinnern Sie sich noch an Ihr erstes Casio-Basic Programm aus Kapitel 2?

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINT "Hallo Casio! ";
20 GOTO 10
```

Hier wird in Zeile 10 über den PRINT-Befehl der Text "Hallo Casio! " ausgegeben. Das abschließende Semikolon verhindert eine Programmunterbrechung. Und in Zeile 20 führt die GOTO-Anweisung dazu, dass das Programm an der Zeile 10 fortgesetzt wird. Es läuft damit endlos, bis Sie die **BRK** Taste drücken, um dem Spuck ein Ende zu bereiten.

### 8.2 Logische Ausdrücke

Um Entscheidungen zu treffen, müssen Sie zuerst noch die möglichen Logischen Ausdrücke kennen, welche zur Entscheidungsfindung verwendet werden können. Ein Logischer Ausdruck kann wahr (-1) oder falsch (0) werden.

Da gibt es zunächst einmal die einfachen Vergleiche, welche mittels 6 verschiedener Vergleichsoperatoren gebildet werden.

Dann gibt es noch 4 Logische Operatoren, die ebenfalls eingesetzt werden können, um Logische Ausdrücke zu verbinden.



**Hinweis**

Es gilt auch hier dieselbe Prioritätsfolge, wie sie bereits im Kapitel 5.3 beschrieben wurde.

Aber sehen Sie selbst...

## 8.2.1 Vergleichsoperatoren

Ein Logischer Ausdruck verbindet Werte oder Variable mit Vergleichsoperatoren. Hier eine vollständige Liste aller möglichen Vergleichsoperatoren:

=		gleich
<		kleiner
>		größer
<>	><	ungleich
<=	=<	kleiner gleich
>=	=>	größer gleich

Vergleichsoperatoren können nur ausgeführt werden, wenn beide Operanden entweder Zeichenfolgen oder numerische Werte sind.

Bei Zeichenfolgen werden die Zeichencodes eine nach der anderen verglichen, beginnend am Anfang der Zeichenfolge. D.h. die erste Position von Zeichenfolge A wird mit der ersten Position von Zeichenfolge B verglichen, die zweite Position von Zeichenfolge A wird mit der zweiten Position von Zeichenfolge B usw. Das Vergleichsergebnis beruht auf dem ersten gefundenen Unterschied zwischen den Vergleichsfolgen, unabhängig von der Länge der Zeichenfolgen, die verglichen werden.



### Beispiel

Nr.	Ausdruck	Beschreibung
1	A=1	Wenn die Variable A den Wert 1 besitzt, dann ist dieser Ausdruck wahr, ansonsten ist er falsch.
2	A<B+1	Wenn der Wert der Variablen A kleiner ist als 1 plus der Wert der Variablen B, dann ist dieser Ausdruck wahr, ansonsten ist er falsch.
3	A>B	Wenn der Wert der Variablen A größer ist als der Wert der Variablen B, dann ist dieser Ausdruck wahr, ansonsten ist er falsch.
4	A\$<>B\$	Wenn die Zeichenfolge der Variablen A\$ ungleich der Zeichenfolge von Variable B\$ ist, dann ist dieser Ausdruck wahr, ansonsten ist er falsch.
5	A\$<="TEST"	Wenn die Zeichenfolge der Variablen A\$ kleiner oder gleich „TEST“, dann ist dieser Ausdruck wahr, ansonsten ist er falsch.
6	"Test">=LEFT\$(A\$,4)	Die Funktion <b>LEFT\$(A\$,4)</b> liefert die ersten vier Zeichen von links aus der Zeichenfolge der Variablen A\$. Dieser Ausdruck ist somit dann wahr, wenn diese vier Zeichen kleiner oder gleich der Zeichenfolge „Test“ sind, ansonsten ist er falsch..
7	ABS p<u*u*1e-10	Die Funktion <b>ABS p</b> liefert den Absolutwert der Variablen p. Dieser Ausdruck ist somit dann wahr, wenn dieser Absolutwert kleiner als der Wert der Variablen u im Quadrat mal 0.0000000001 ist, ansonsten ist er falsch..
8	LEN(f\$)<25	Die Funktion <b>LEN(f\$)</b> liefert die Anzahl von Zeichen der Zeichenfolge f\$. Dieser Ausdruck ist somit dann wahr, wenn diese Anzahl Zeichen kleiner als 25 ist, ansonsten ist er falsch..

### 8.2.2 Logische Operatoren

Sie dürfen Logische Ausdrücke auch verbinden, indem Sie einen der vier möglichen logischen Operatoren verwenden:

NOT	Negation
AND	Und-Verknüpfung
OR	Oder-Verknüpfung
XOR	Exklusiv-Oder-Verknüpfung



**Beispiel**

Nr.	Ausdruck	Beschreibung
1	NOT A	Dieser Ausdruck ist wahr, wenn der Inhalt der Variablen A gleich 0 ist, ansonsten ist er falsch.
2	A>10 AND A<20	Dieser Ausdruck ist wahr, wenn der Inhalt der Variablen A größer 10 und kleiner 20 ist, ansonsten ist er falsch.
3	A OR B	Dieser Ausdruck ist wahr, wenn das Ergebnis der Logischen ODER Verknüpfung von den Variablen A mit B ungleich 0 ist, ansonsten ist er falsch. Mit anderen Worten: Nur wenn A=0 und B=0 ist, wird der Ausdruck falsch.
4	A=1 AND B=2 XOR A=2 AND B=3	Bedingt durch die Rangreihenfolge der Operatoren werden zunächst die Und-Operationen und anschließend die XOR-Operation ausgewertet.
5	A\$="J" OR A\$="j"	Dieser Ausdruck ist wahr, wenn der Inhalt der Variablen A\$ gleich „J“ oder „j“ ist, ansonsten ist er falsch.
6	IFa(0)<14 ORa(0)>127	Dieser Ausdruck ist wahr, wenn der Inhalt der Feldvariablen a(0) kleiner 14 oder größer 127 ist, ansonsten ist er falsch.
7	x<2 ORx>=1e10 ORFRACx<>0	Die Funktion <b>FRAC x</b> liefert den Nachkommawert der Variablen x. Dieser Ausdruck ist somit dann wahr, wenn dieser Nachkommawert ungleich 0 ist oder x kleiner 2 ist oder x größer gleich 10000000000, ansonsten ist er falsch.
8	r<>0 ANDABSr<1e-90	Die Funktion <b>ABS r</b> liefert den Absolutwert von r. Dieser Ausdruck ist somit dann wahr, wenn r ungleich 0 ist und der Absolutwert von r kleiner 1E-90, ansonsten ist er falsch.

### 8.3 Die IF-Anweisung

Die IF-Anweisung führt die THEN-Anweisung oder GOTO-Anweisung aus, wenn die spezifizierte Bedingung erfüllt wird. Die ELSE-Anweisung wird ausgeführt, wenn die spezifizierte Bedingung nicht erfüllt wird. Die gesamte IF ~ THEN ~ ELSE Anweisung muss in einer einzigen Programmzeile untergebracht werden.

Nach dem IF muss entweder eine Bedingung oder ein numerischer Ausdruck angegeben werden. Eine Bedingung kann entweder ein einfacher Vergleich sein (z.B.  $A=1$  oder  $A<>B$  oder  $A\>B$ ), oder ein Logischer Ausdruck (z.B.  $A>10$  AND  $A\leq 20$  OR  $B\neq ""$ )

IF	Bedingung Num. Ausdruck	THEN	Anweisung Verzweigungsziel
----	----------------------------	------	-------------------------------

bzw.

IF	Bedingung Num. Ausdruck	GOTO	Anweisung Verzweigungsziel
----	----------------------------	------	-------------------------------

Im Folgenden erkläre ich Ihnen alle möglichen Varianten.

#### 8.3.1 IF ~ THEN

Bei diesem Beispiel wird nur dann das Wort „zehn“ ausgegeben, wenn die eingegebene Zahl genau 10 ist.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8888 DEFM PRT TR STOP
10 PRINT "Zahl eingeben";
20 INPUT ZAHL
30 IF ZAHL=10 THEN PRINT "zehn"
```



#### Achtung

Beim Format `IF A THEN` muss mindestens ein Leerzeichen zwischen dem numerischen Ausdruck und dem THEN stehen!

#### 8.3.2 IF ~ THEN ~ ELSE

Bei diesem Beispiel wird das Wort „negativ“ ausgegeben, wenn die Eingabe eine negative Zahl ist. Andernfalls wird das Wort „positiv“ ausgegeben.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8888 DEFM PRT TR STOP
10 PRINT "Zahl eingeben";
20 INPUT ZAHL
30 IF ZAHL<0 THEN PRINT
"negativ" ELSE PRINT "positiv"
```

### 8.3.3 IF ~ GOTO

Mittels des IF ~ GOTO Konstrukts, wird direkt zur angegebenen Programmzeile gesprungen, wenn die IF-Bedingung erfüllt ist. Dadurch kann ein Programmteil wiederholt ausgeführt werden. In diesem Zusammenhang spricht man von einer Programmschleife.

Bei diesem Beispiel wird fortlaufend die Summe der bis dahin eingegeben Zahlen ausgegeben.

Nur bei der Eingabe der Zahl 0 wird das Programm beendet.

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
5 S=0
10 PRINT "Zahl eingeben";
20 INPUT ZAHL
30 SUMME=SUMME+ZAHL
40 PRINT "Summe=";SUMME;
50 IF ZAHL GOTO 10

```

**Achtung**

Beim Format **IF A GOTO** muss mindestens ein Leerzeichen zwischen dem numerischen Ausdruck und dem GOTO stehen!

### 8.3.4 IF ~ GOTO ~ ELSE

Bei diesem Beispiel wird bei Eingabe der Zahl 1, 2 oder 3 das Wort „OK“ ausgegeben.

Bei jeder anderen Zahlen-eingabe wird die Eingabe wiederholt.

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
10 PRINT "Zahl von 1 bis 3
einsetzen";
20 INPUT ZAHL
30 IF ZAHL<1 OR ZAHL>3 GOTO 10
ELSE PRINT "OK"

```

## 8.4 Ausführbare Befehle der IF-Anweisung

Rufen Sie sich die Definition der IF-Anweisung noch einmal ins Gedächtnis zurück:

IF	Bedingung Num. Ausdruck	THEN	Anweisung Verzweigungsziel
----	----------------------------	------	-------------------------------

Nachdem Sie jetzt mit den Bedingungen und deren Logische Ausdrücke vertraut sind, will ich Ihnen die rechte Seite dieser Definition erklären.

### 8.4.1 Anweisung

Eine Anweisung ist ein ausführbarer Befehl. Sie kann eine Zuweisung-, eine INPUT- oder eine PRINT-Anweisung sein. Darf jedoch keine andere IF-Anweisung oder ein Kommando wie etwas REM, NEW oder LIST sein. Erinnern Sie sich an das Zeilenformat einer Programmzeile?



**Hinweis**

Anweisungen können durch einen Doppelpunkt (:) voneinander getrennt sein!

Deshalb können Sie durch die Verwendung eines Doppelpunktes (:) sogar mehrere ausführbarer Befehle nacheinander schreiben.

Bei diesem Beispiel wird der Bildschirm gelöscht, das Wort „zehn“ ausgegeben, die Variable EINGABE mit dem Wert 10 belegt und in die Programmzeile 230 gesprungen, falls die Bedingung wahr ist.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
30 IF ZAHL=10 THEN CLS: PRINT
"zehn": EINGABE=ZAHL: GOTO 230
```

### 8.4.2 Verzweigungsziel

Beim Verzweigungsziel handelt es sich normalerweise um eine Zeilennummer. Aber das CASIO-Basic lässt auch Numerische Variable zu. Deren Inhalt definiert dann die Programmzeile.

Die Zeilennummer zu welche die GOTO-Anweisung springt, wird in nebenstehendem Beispiel berechnet aus dem Variableninhalt von ZAHL (welcher 1, 2 oder 3 sein kann) minus 1 mal 50 plus 100.

Als Ergebnis wird somit zur Zeilennummer 100, 150 oder 200 gesprungen.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINT "1, 2 oder 3";
20 INPUT ZAHL
30 IF ZAHL<1 OR ZAHL>3 GOTO 10
ELSE CLS: GOTO (ZAHL-1)*50+100
100 PRINT "Erster"
110 GOTO 10
150 PRINT "Zweiter"
160 GOTO 10
200 PRINT "Dritter"
210 GOTO 10
```

Für dieses Programmbeispiel könnte man auch die **ON GOTO**-Anweisung verwenden.

## 8.5 Eine Rechenübung

### 8.5.1 Einführung

Mit Hilfe der neu erlernten Fähigkeiten zeige ich Ihnen ein Programm, welches einfache Rechenübungen durchführt. Es besteht aus einem Menü, bei dem der Anwender zwischen den Rechenübungen Addition, Subtraktion, Multiplikation und Division auswählen kann. Sobald der Anwender seine Auswahl getroffen hat, wird er aufgefordert Rechenübungen durchzuführen. Die Übungen wiederholen sich, solange bis der Anwender eine 0 eingibt.

### 8.5.2 Neue Funktionen für die Rechenübung

Bei dem Programm werden fast nur bereits bekannte BASIC-Anweisungen eingesetzt. Nur die Funktionen **RAN#** und **INT** sind für Sie neu und müssen noch erklärt werden, bevor wir uns das Programmlisting ansehen.

#### 8.5.2.1 Zufallswerte erzeugen

Die Funktion **RAN#** erzeugt einen Zufallswert im Bereich vom 0 bis 1.

Wenn Sie die Anweisung **PRINT RAN#** mehrmals hintereinander ausführen,

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT RAN#
0.7938223568
```

Wird jedes Mal ein anderes Ergebnis ausgegeben. Der kleinste Zufallswert ist 0, der Größte ist 0.999999999.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT RAN#
0.0132562856
```

#### 8.5.2.2 Beliebige große Zufallswerte

Durch eine einfache Multiplikation können Sie auch beliebig größere Zufallszahlen erzeugen.

Diese Anweisung erzeugt Zufallszahlen von 0 bis 99.999999999

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT RAN#*100
13.5336023954
```

#### 8.5.2.3 Ganzzahlige Zufallswerte

Um einen ganzzahligen Zufallswert zu erzeugen können Sie sich die Funktion **INT** zu Hilfe nehmen. Die **INT**-Funktion ermittelt den ganzzahligen Anteil des Arguments.

Diese Anweisung erzeugt Zufallszahlen von 0 bis 99

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT INT(RAN#*100)
56
```

### 8.5.2.4 Zufallswerte von ... bis ...

Meist werden jedoch Zufallswerte von einem bestimmten Startwert bis zu einem Endwert benötigt. Der Startwert kann einer Zufallszahl einfach aufaddiert werden.

Diese Anweisung erzeugt Zufallszahlen von 1 bis 6.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
PRINT INT(RAN#*6)+1
2
```

### 8.5.3 Programmversion 1.00

So, mit dem jetzigen Wissen können wir nun endlich starten. Bei der ersten Version des Programms ist es wichtig, dass es einfach aufgebaut ist und richtig funktioniert. Das Programm besteht aus 3 Teilen:

1. Programmkopf
  - a. Programmname
  - b. Version, Datum
  - c. Autor
2. Auswahlmenü
  - a. Menü-Anzeige
  - b. Menü-Eingabe
  - c. Eingabeauswertung
3. Berechnungen
  - a. Addition
  - b. Subtraktion
  - c. Multiplikation
  - d. Division

#### 8.5.3.1 Programmkopf (Zeile 100-140)

Durch einleitende Kommentare werden der Programmname, Version, Datum und Autor festgehalten.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
100 REM -----
110 REM Eine Rechenuebung
120 REM Version 1.00 vom 04.08.2009
130 REM Autor: Manfred Becker
140 REM -----
```

#### 8.5.3.2 Auswahlmenü (Zeile 200-283)

Die Anzeige des Auswahlmenüs erfolgt in Zeile 210-230. Mit LOCATE wird die Position für das anschließende PRINT

bestimmt. Die PRINT-Anweisung muss mit einem Semikolon (;) abgeschlossen werden, um eine Programmunterbrechung zu vermeiden.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
200 REM Auswahlmenue
210 CLS
220 LOCATE 0,0: PRINT"0=Ende 1=Addition 2=Subtraktion";
230 LOCATE 0,1: PRINT"3=Multiplikation 4=Division";
```

Die User-Eingabe erfolgt mittels INPUT-Befehl in Zeile 240.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
240 INPUT auswahl
```

Da zur Eingabe nur Werte von 0 bis 4 zulässig sind, wird bei

anderen Eingaben eine entsprechende Fehlermeldung angezeigt. Danach muss die Eingabe wiederholt werden. Das wird in Zeile 250 realisiert.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
250 IF auswahl<0 OR auswahl>4 THEN PRINT"Falsche
Eingabe!": GOTO 200
```

Mit der Eingabe einer 0 wird das Programm beendet.

Beachten Sie den END-Befehl. Dadurch wird die Programmausführung beendet.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
260 IF auswahl=0 THEN PRINT"Auf wiedersehen!": END
```

Nun erfolgt zu jeder Auswahl der Sprung zu der entsprechenden Programmzeile. Zuvor werden noch die zwei

Variable *richtig* und *falsch* initialisiert (mit einem definierten Wert vorbelegt.).

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
270 richtig=0: falsch=0
280 IF auswahl=1 GOTO 300
281 IF auswahl=2 GOTO 400
282 IF auswahl=3 GOTO 500
283 IF auswahl=4 GOTO 600
```

### 8.5.3.3 Berechnung Addition (Zeile 300-380)

In diesem Programmabschnitt (Zeile 300-380) werden Übungen zur Addition abgehandelt. Zunächst wird das Display gelöscht (Zeile 310), und die Überschrift (incl. Angabe zu richtig bzw. falsch gelöste Übungen) angezeigt (Zeile 320).

Die Zufallswerte für die zwei Summanden werden in Zeile 330 und 340 bestimmt. In Zeile 350 wird die Aufgabenstellung ausgegeben. Die User-Eingabe erfolgt in Zeile 360. Bei der Eingabe einer 0 wird sofort zurück gesprungen ins Hauptprogramm. Alle anderen Eingabewerte werden in Zeile 370 mit dem Sollwert verglichen. Falls die Eingabe OK war, wird die entsprechende OK-Meldung ausgegeben. Die Variable für richtig gelöste Aufgaben wird inkrementiert. Falls ein Rechenfehler entdeckt wurde, wird ebenfalls eine entsprechende Fehler-Meldung ausgegeben (incl. Lösung). Die Variable für falsch gelöste Aufgaben wird inkrementiert. In Zeile 480 bewirkt der Sprung zur Zeile 400 eine Wiederholung der Übung.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
300 REM Addition
310 CLS
320 LOCATE 0,0: PRINT"Addition: ";richtig;"richtig
";falsch;"falsch";
330 zahl2=INT(RAN#*10)+1
340 zahl1=INT(RAN#*10)+1
350 LOCATE 0,1:PRINT"Wieviele ist";zahl1;"+";zahl2;
360 INPUT einsabe: IF einsabe=0 GOTO 200
370 IF einsabe=zahl1+zahl2 THEN richtig=richtig+1:
PRINT"Ihre Einsabe war richtig!" ELSE falsch=falsch+1:
PRINT"Fehler! Richtig waere";zahl1+zahl2
380 GOTO 300
```

### 8.5.3.4 Berechnung Subtraktion (Zeile 400-480)

Die Übungen zur Subtraktion verlaufen fast identisch zu der Addition. Die fett markierten Stellen weisen die jeweiligen Unterschiede auf. In Zeile 440 sorgen wir mit der Addition von *zahl2* dafür, dass keine negativen Ergebnisse entstehen können.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
400 REM Subtraktion
410 CLS
420 LOCATE 0,0: PRINT"Subtraktion: ";richtig;"richtig
";falsch;"falsch";
430 zahl2=INT(RAN#*10)+1
440 zahl1=zahl2 + INT(RAN#*10)+1
450 LOCATE 0,1:PRINT"Wieviele ist";zahl1;"-";zahl2;
460 INPUT einsabe: IF einsabe=0 GOTO 200
470 IF einsabe=zahl1-zahl2 THEN richtig=richtig+1:
PRINT"Ihre Einsabe war richtig!" ELSE falsch=falsch+1:
PRINT"Fehler! Richtig waere";zahl1-zahl2
480 GOTO 400
```

### 8.5.3.5 Berechnung Multiplikation (Zeile 500-580)

Die Übungen zur Multiplikation verlaufen fast identisch zu der Addition. Die fett markierten Stellen weisen die jeweiligen Unterschiede auf.

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
500 REM Multiplikation
510 CLS
520 LOCATE 0,0: PRINT"Multiplikation: ";richtig;"richtig
";falsch;"falsch";
530 zahl2=INT(RAN#*10)+1
540 zahl1=INT(RAN#*10)+1
550 LOCATE 0,1:PRINT"Wieviele ist";zahl1;"*";zahl2;
560 INPUT einsabe: IF einsabe=0 GOTO 200
570 IF einsabe=zahl1*zahl2 THEN richtig=richtig+1:
PRINT"Ihre Einsabe war richtig!" ELSE falsch=falsch+1:
PRINT"Fehler! Richtig waere";zahl1*zahl2
580 GOTO 500

```

### 8.5.3.6 Berechnung Division (Zeile 600-680)

Die Übungen zur Division verlaufen fast identisch zu der Addition. Die fett markierten Stellen weisen die jeweiligen Unterschiede auf. In Zeile 640 sorgen wir mit der Multiplikation von *zahl2* dafür, dass nur ganzzahlige Ergebnisse entstehen können.

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
600 REM Division
610 CLS
620 LOCATE 0,0: PRINT"Division: ";richtig;"richtig
";falsch;"falsch";
630 zahl2=INT(RAN#*10)+1
640 zahl1=zahl2 * (INT(RAN#*10)+1)
650 LOCATE 0,1:PRINT"Wieviele ist";zahl1;" / ";zahl2;
660 INPUT einsabe: IF einsabe=0 GOTO 200
670 IF einsabe=zahl1/zahl2 THEN richtig=richtig+1:
PRINT"Ihre Einsabe war richtig!" ELSE falsch=falsch+1:
PRINT"Fehler! Richtig waere";zahl1/zahl2
680 GOTO 600

```

So, das war's auch schon. War doch gar nicht so schwierig, oder? Ein komplettes Programmlisting finden Sie im nächsten Abschnitt.



#### Hinweis

Übrigens können Sie auf meiner Homepage alle Listings zu diesem Tutorial herunterladen. Das erspart Ihnen so manche Tipparbeit.

### 8.5.3.7 Das Programm-Listing

Dieses Programm sieht eindrucksvoll aus, ist aber in Wirklichkeit ganz einfach. Lassen Sie sich von dem „großen“ Programmumfang nicht abschrecken. Ich hoffe, Sie können durch meine einleitende Beschreibung das Programm vollständig verstehen.

#### Listing 2

Rechenübung.100.cas

(53 Programmzeilen)

```

100 REM -----
110 REM Eine Rechenuebung
120 REM Version 1.00 vom 04.08.2009
130 REM Autor: Manfred Becker
140 REM -----
200 REM Auswahlmenue
210 CLS
220 LOCATE 0,0: PRINT"0=Ende 1=Addition 2=Subtraktion";
230 LOCATE 0,1: PRINT"3=Multiplikation 4=Division";
240 INPUT auswahl
250 IF auswahl<0 OR auswahl>4 THEN PRINT"Falsche Einsabe!": GOTO 200
260 IF auswahl=0 THEN PRINT"Auf wiedersehen!": END
270 richtig=0: falsch=0
280 IF auswahl=1 GOTO 300
281 IF auswahl=2 GOTO 400
282 IF auswahl=3 GOTO 500
283 IF auswahl=4 GOTO 600
300 REM Addition
310 CLS
320 LOCATE 0,0: PRINT"Addition: ";richtig;"richtig ";falsch;"falsch";
330 zahl2=INT(RAN#*10)+1
340 zahl1=INT(RAN#*10)+1
350 LOCATE 0,1:PRINT"Wieviel ist";zahl1;"+";zahl2;
360 INPUT einsabe: IF einsabe=0 GOTO 200
370 IF einsabe=zahl1+zahl2 THEN richtig=richtig+1: PRINT"Ihre Einsabe war richtig!"
ELSE falsch=falsch+1: PRINT"Fehler! Richtig waere";zahl1+zahl2
380 GOTO 300
400 REM Subtraktion
410 CLS
420 LOCATE 0,0: PRINT"Subtraktion: ";richtig;"richtig ";falsch;"falsch";
430 zahl2=INT(RAN#*10)+1
440 zahl1=zahl2 + INT(RAN#*10)+1
450 LOCATE 0,1:PRINT"Wieviel ist";zahl1;"-";zahl2;
460 INPUT einsabe: IF einsabe=0 GOTO 200
470 IF einsabe=zahl1-zahl2 THEN richtig=richtig+1: PRINT"Ihre Einsabe war richtig!"
ELSE falsch=falsch+1: PRINT"Fehler! Richtig waere";zahl1-zahl2
480 GOTO 400
500 REM Multiplikation
510 CLS
520 LOCATE 0,0: PRINT"Multiplikation: ";richtig;"richtig ";falsch;"falsch";
530 zahl2=INT(RAN#*10)+1
540 zahl1=INT(RAN#*10)+1
550 LOCATE 0,1:PRINT"Wieviel ist";zahl1;"*";zahl2;
560 INPUT einsabe: IF einsabe=0 GOTO 200
570 IF einsabe=zahl1*zahl2 THEN richtig=richtig+1: PRINT"Ihre Einsabe war richtig!"
ELSE falsch=falsch+1: PRINT"Fehler! Richtig waere";zahl1*zahl2
580 GOTO 500
600 REM Division
610 CLS
620 LOCATE 0,0: PRINT"Division: ";richtig;"richtig ";falsch;"falsch";
630 zahl2=INT(RAN#*10)+1
640 zahl1=zahl2 * (INT(RAN#*10)+1)
650 LOCATE 0,1:PRINT"Wieviel ist";zahl1;" / ";zahl2;
660 INPUT einsabe: IF einsabe=0 GOTO 200
670 IF einsabe=zahl1/zahl2 THEN richtig=richtig+1: PRINT"Ihre Einsabe war richtig!"
ELSE falsch=falsch+1: PRINT"Fehler! Richtig waere";zahl1/zahl2
680 GOTO 600

```

### 8.5.4 Eine verbesserte Version 1.01

Kaum ist unser erstes Programm fertig, denkt „der richtige Programmierer“ über etwaige Programmverbesserungen nach. Insbesondere ist vielleicht auch Ihnen aufgefallen, dass die Programmabschnitte für die Berechnungen sich sehr ähnlich sehen.



Die Frage wäre, ob man hier nicht etwas vereinfachen könnte. Und tatsächlich bietet das CASIO-Basic eine Möglichkeit an gleichartige Programmabschnitte als Unterprogramme (engl. Subroutines) auszulegen.

Mehr dazu im folgenden Unterkapitel...

#### 8.5.4.1 Unterprogramme (eng. Subroutines)

Bisher bestanden Ihre Programme immer nur aus einem Haupt-Programm. Dieses fängt mit der ersten Programmzeile an, und endet mit der Letzten. Dort kann (muss aber nicht) der BASIC-Befehl END stehen.

Hier ein typisches Haupt-Programm.

```
CAPS S CAL BASIC DEG/GRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
100 REM Hauptprogramm
110 PRINT "Hallo"
120 END
```

Allerdings dürfen Sie nach der Programmzeile mit dem **END**-Befehl weitere Programmzeilen anfügen. Diese werden nicht ausgeführt, denn mit dem END-Befehl endet die Abarbeitung bedingungslos. Allerdings können Sie mit dem **GOSUB**-Befehl zu solchen Programmzeilen springen. Ihr Unterprogramm muss lediglich mit dem **RETURN**-Befehl abgeschlossen sein!

Hier ein typisches Haupt-Programm mit einem Unterprogramm ab Zeile 200.

```
CAPS S CAL BASIC DEG/GRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
100 REM Hauptprogramm
110 PRINT "Hallo"
120 r=10
130 GOSUB 200
140 r=20
150 GOSUB 200
160 END
200 REM Unterprogramm
210 A=PI*r^2
220 PRINT "Die Flaeche eines Kreises mit dem Radius":r:
230 PRINT " betraest":A
230 RETURN
```

Im Hauptprogramm wird das Unterprogramm an zwei Stellen mit GOSUB aufgerufen (Zeile 130 und 150).

Damit dürfte Ihnen der Vorteil von solchen Unterprogrammen klar werden. Programmabschnitte die an mehreren Stellen des Programms identisch (oder fast identisch) vorkommen, können so ausgelagert werden, und müssen nur einmalig programmiert werden. Der Programmumfang nimmt ab, die Wartung wird vereinfacht und die Fehleranfälligkeit wird verringert.

Unterprogramme dürfen mittels GOSUB in weitere Unterprogramme springen. Aber achten Sie bitte darauf, dass Sie niemals aus einem Unterprogramm mit dem GOTO-Befehl zurück in das Hauptprogramm springen. Das führt früher oder später zu einem Programm-Absturz. Die Fehlersuche erweist sich in solchen Fällen äußerst schwierig.

### 8.5.4.2 Das Unterprogramm

Mit diesem Wissen, programmieren wir unser erstes Unterprogramm ab Programmzeile 700. Dabei sorgen wir dafür, dass alle Werte die sich bei den verschiedenen Berechnungsarten unterscheiden, variabel ausgeführt werden.

In diesem Programmabschnitt (Zeile 700-810) befindet sich unser Unterprogramm. Die fett markierten Stellen zeigen die Unterschiede zum vorhergehenden Berechnungsabschnitt.

Zum einen wird die Variable **titel\$** verwendet. Sie beinhaltet je nach Rechenart die entsprechende Überschrift.

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
700 REM Unterprogramm zur Berechnung
710 CLS
720 LOCATE 0,0: PRINT titel$:" ":richtig;"richtig"
":falsch;"falsch";
730 zahl1=INT(RAN#*10)+1
740 IF op$="+" THEN zahl1=INT(RAN#*10)+1:
ergebnis=zahl1+zahl2
750 IF op$="-" THEN zahl1=zahl2 + INT(RAN#*10)+1:
ergebnis=zahl1-zahl2
760 IF op$="*" THEN zahl1=INT(RAN#*10)+1:
ergebnis=zahl1*zahl2
770 IF op$="/" THEN zahl1=zahl2 * (INT(RAN#*10)+1):
ergebnis=zahl1/zahl2
780 LOCATE 0,1:PRINT"Wieviel ist";zahl1;op$;zahl2;
790 INPUT einsabe: IF einsabe=0 THEN RETURN
800 IF einsabe=ergebnis THEN richtig=richtig+1:
PRINT"Ihre Einsabe war richtig!" ELSE falsch=falsch+1:
PRINT"Fehler! Richtig waere";ergebnis
810 RETURN

```

Dann gibt es die Variable **op\$**, welche den Operanden der ausgewählten Rechenart beinhaltet. Beachten Sie bitte, dass Abhängig von der Rechenart, die zweite Zufallszahl gebildet wird. Und schließlich wird in der Variablen **ergebnis** das Rechenergebnis gespeichert. So kann der Vergleich zur Benutzereingabe realisiert werden. Außerdem wird das Ergebnis im Fehlerfall angezeigt.

Sie dürfen im Unterprogramm GOTO verwenden, um an eine andere Stelle des Unterprogramms zu gelangen, aber der Rücksprung erfolgt immer mit RETURN (Zeile 790 u. 810).

### 8.5.4.3 Der Aufruf des Unterprogramms

Jetzt, nachdem das Unterprogramm fertig ist, kümmern wir uns um den Aufruf dieses Unterprogramms. Erinnern Sie sich an die vier Rechenmodule?

- Berechnung Addition (Zeile 300-380)
- Berechnung Subtraktion (Zeile 400-480)
- Berechnung Multiplikation (Zeile 500-580)
- Berechnung Division (Zeile 600-680)

Diese können jetzt erheblich vereinfacht werden, denn wir benötigen nun je Modul nur noch folgende 3 Programmschritte:

- Variable entsprechend dem Rechenmodul setzen  
Hiebei geht es um die Variablen **op\$** und **titel\$**
- Aufruf des Unterprogramms  
Das erledigt der Basic-Befehl GOSUB 700
- Abfrage der Eingabe-Variablen  
Hiervon ist abhängig, ob weitergerechnet oder beendet werden soll.

Schauen Sie sich die Veränderung einmal an:

Die fett markierten Stellen zeigen die Unterschiede.

Zuvor benötigten wir für die vier Rechenmodule 36 Programmzeilen. Nun sind diese auf 16 Zeilen zusammengeschumpft.

Außerdem ist dieser Programmabschnitt viel Übersichtlicher geworden.

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
300 REM Addition
310 op$="+": titel$="Addition"
320 GOSUB 700
330 IF einsabe=0 GOTO 200 ELSE 300
400 REM Subtraktion
410 op$="-": titel$="Subtraktion"
420 GOSUB 700
430 IF einsabe=0 GOTO 200 ELSE 400
500 REM Multiplikation
510 op$="*": titel$="Multiplikation"
520 GOSUB 700
530 IF einsabe=0 GOTO 200 ELSE 500
600 REM Division
610 op$="/": titel$="Division"
620 GOSUB 700
630 IF einsabe=0 GOTO 200 ELSE 600

```

### 8.5.4.4 Das Programm-Listing

Hier nun die kompaktere Version von unserer Rechenübung. Dank Unterprogramm benötigen wir nur noch 45 statt 53 Programmzeilen.

#### Listing 3

Rechenübung.101.cas

(45 Programmzeilen)

```

100 REM -----
110 REM Eine Rechenuebung
120 REM Version 1.01 vom 04.08.2009
130 REM Autor: Manfred Becker
140 REM -----
200 REM Auswahlmenue
210 CLS
220 LOCATE 0,0: PRINT"0=Ende 1=Addition 2=Subtraktion";
230 LOCATE 0,1: PRINT"3=Multiplikation 4=Division";
240 INPUT auswahl
250 IF auswahl<0 OR auswahl>4 THEN PRINT"Falsche Einsabe!": GOTO 200
260 IF auswahl=0 THEN PRINT"Auf wiedersehen!": END
270 richtig=0: falsch=0
280 IF auswahl=1 GOTO 300
281 IF auswahl=2 GOTO 400
282 IF auswahl=3 GOTO 500
283 IF auswahl=4 GOTO 600
300 REM Addition
310 op$="+": titel$="Addition"
320 GOSUB 700
330 IF einsabe=0 GOTO 200 ELSE 300
400 REM Subtraktion
410 op$="-": titel$="Subtraktion"
420 GOSUB 700
430 IF einsabe=0 GOTO 200 ELSE 400
500 REM Multiplikation
510 op$="*": titel$="Multiplikation"
520 GOSUB 700
530 IF einsabe=0 GOTO 200 ELSE 500
600 REM Division
610 op$="/": titel$="Division"
620 GOSUB 700
630 IF einsabe=0 GOTO 200 ELSE 600
700 REM Unterprogramm zur Berechnung
710 CLS
720 LOCATE 0,0: PRINT titel$:" ":richtig:"richtig ":falsch:"falsch";
730 zahl2=INT(RAN#*10)+1
740 IF op$="+" THEN zahl1=INT(RAN#*10)+1: ergebnis=zahl1+zahl2
750 IF op$="-" THEN zahl1=zahl2 + INT(RAN#*10)+1: ergebnis=zahl1-zahl2
760 IF op$="*" THEN zahl1=INT(RAN#*10)+1: ergebnis=zahl1*zahl2
770 IF op$="/" THEN zahl1=zahl2 * (INT(RAN#*10)+1): ergebnis=zahl1/zahl2
780 LOCATE 0,1:PRINT"Wieviel ist";zahl1;op$;zahl2;
790 INPUT einsabe: IF einsabe=0 THEN RETURN
800 IF einsabe=ergebnis THEN richtig=richtig+1: PRINT"Ihre Einsabe war richtig!" ELSE
falsch=falsch+1: PRINT"Fehler! Richtig waere";ergebnis
810 RETURN

```

### 8.5.5 Eine neuer Ansatz Version 2.00

So ganz zufrieden bin ich aber auch mit der neuen Version auch noch nicht. Sie funktioniert zwar genauso wie die Erste, ist etwas kompakter und leichter zu verstehen, aber mich stört noch folgende Programmlogik:

Hier erfolgt der Sprung in die entsprechenden Programmabschnitte.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
280 IF auswahl=1 GOTO 300
281 IF auswahl=2 GOTO 400
282 IF auswahl=3 GOTO 500
283 IF auswahl=4 GOTO 600
```

Das war bei der Version 1.00 so noch in Ordnung, aber jetzt ist doch der einzige Unterschied in diesen Programmabschnitten die Zuweisung der Variable. Der Aufruf vom Unterprogramm und die anschließende Entscheidung ob weitergerechnet werden soll, sind doch bei allen absolut identisch.

Wieso setzen wir dann nicht die Zuweisung direkt an diese Auswahl, rufen das Unterprogramm und prüfen dann ob weitergerechnet werden soll. Damit würden die Programmabschnitte von Zeile 300 bis Zeile 630 komplett wegfallen!

#### 8.5.5.1 Das Programm-Listing

Hier die Umsetzung von der Idee aus dem vorhergehenden Kapitel. Gleichzeitig wurde auch ab Programmzeile 280 neu durchnummeriert. Das Ergebnis kann sich sehen lassen. Jetzt sind es nur noch 30 Programmzeilen, bei absolut gleicher Funktionalität!

#### Listing 4

Rechenübung.200.cas

(30 Programmzeilen)

```
100 REM -----
110 REM Eine Rechenuebung
120 REM Version 2.00 vom 04.08.2009
130 REM Autor: Manfred Becker
140 REM -----
200 REM Auswahlmenue
210 CLS
220 LOCATE 0,0: PRINT"0=Ende 1=Addition 2=Subtraktion";
230 LOCATE 0,1: PRINT"3=Multiplikation 4=Division";
240 INPUT auswahl
250 IF auswahl<0 OR auswahl>4 THEN PRINT"Falsche Einsabe!": GOTO 200
260 IF auswahl=0 THEN PRINT"Auf wiedersehen!": END
270 richtig=0: falsch=0
280 IF auswahl=1 THEN op$="+": titel$="Addition"
290 IF auswahl=2 THEN op$="-": titel$="Subtraktion"
300 IF auswahl=3 THEN op$="*": titel$="Multiplikation"
310 IF auswahl=4 THEN op$="/": titel$="Division"
320 GOSUB 400: IF einsabe=0 GOTO 200 ELSE 320
400 REM Unterprogramm zur Berechnung
410 CLS
420 LOCATE 0,0: PRINT titel$; " "; richtig;"richtig "; falsch;"falsch";
430 zahl2=INT(RAN#*10)+1
440 IF op$="+" THEN zahl1=INT(RAN#*10)+1: ergebnis=zahl1+zahl2
450 IF op$="-" THEN zahl1=zahl2 + INT(RAN#*10)+1: ergebnis=zahl1-zahl2
460 IF op$="*" THEN zahl1=INT(RAN#*10)+1: ergebnis=zahl1*zahl2
470 IF op$="/" THEN zahl1=zahl2 * (INT(RAN#*10)+1): ergebnis=zahl1/zahl2
480 LOCATE 0,1:PRINT"Wieviel ist";zahl1;op$;zahl2;
490 INPUT einsabe: IF einsabe=0 THEN RETURN
500 IF einsabe=ergebnis THEN richtig=richtig+1: PRINT"Ihre Einsabe war richtig!" ELSE
falsch=falsch+1: PRINT"Fehler! Richtig waere";ergebnis
510 RETURN
```

## 8.5.6 Eine verbesserte Version 2.01

Sie haben es vielleicht schon bemerkt – Ich bin ein Perfektionist. Eine Eigenschaft die für einen Programmierer nicht immer hilfreich ist. Ein Programm muss fehlerfrei funktionieren und verständlich programmiert sein, damit die Wartung einfach bleibt. Das alles erfüllt unsere Version 2.00.



Dennoch ist mir aufgefallen, dass durch unsere Programmvereinfachung unser Unterprogramm nur noch einer einzigen Stelle aufgerufen wird.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
320 GOSUB 400: IF einsabe=0 GOTO 200 ELSE 320
```

Und damit ist es eigentlich gar nicht mehr notwendig ein Unterprogramm zu verwenden.

Aus diesem Grund habe ich die Programmzeile 320 gelöscht, und die RETURN-Anweisungen durch GOTO-Befehle ersetzt.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
490 INPUT einsabe: IF einsabe=0 THEN 200
500 IF einsabe=ergebnis THEN richtig=richtig+1:
PRINT"Ihre Einsabe war richtig!" ELSE falsch=falsch+1:
PRINT"Fehler! Richtig waere":ergebnis
510 GOTO 400
```

So, jetzt kann ich beruhigt schlafen...

### 8.5.6.1 Das Programm-Listing

Das ist sie nun, die „Finale Version“ von dem Programm Rechenübung. Ich hoffe, Sie konnten die Entstehungsgeschichte nachvollziehen. Wie bereits ganz am Anfang erwähnt sollen Sie auch an solchen Dingen teilhaben. Daraus lernen Sie mehr, als wenn ich Ihnen sofort die Ultimative Lösung präsentieren würde.

#### Listing 5

Rechenübung.201.cas

(29 Programmzeilen)

```
100 REM -----
110 REM Eine Rechenuebung
120 REM Version 2.01 vom 04.08.2009
130 REM Autor: Manfred Becker
140 REM -----
200 REM Auswahlmenue
210 CLS
220 LOCATE 0,0: PRINT"0=Ende 1=Addition 2=Subtraktion";
230 LOCATE 0,1: PRINT"3=Multiplikation 4=Division";
240 INPUT auswahl
250 IF auswahl<0 OR auswahl>4 THEN PRINT"Falsche Einsabe!": GOTO 200
260 IF auswahl=0 THEN PRINT"Auf wiedersehen!": END
270 richtig=0: falsch=0
280 IF auswahl=1 THEN op$="+": titel$="Addition"
290 IF auswahl=2 THEN op$="-": titel$="Subtraktion"
300 IF auswahl=3 THEN op$="*": titel$="Multiplikation"
310 IF auswahl=4 THEN op$="/": titel$="Division"
400 REM Berechnung
410 CLS
420 LOCATE 0,0: PRINT titel$:" ":richtig:"richtig ":falsch:"falsch";
430 zahl2=INT(RAN#*10)+1
440 IF op$="+" THEN zahl1=INT(RAN#*10)+1: ergebnis=zahl1+zahl2
450 IF op$="-" THEN zahl1=zahl2 + INT(RAN#*10)+1: ergebnis=zahl1-zahl2
460 IF op$="*" THEN zahl1=INT(RAN#*10)+1: ergebnis=zahl1*zahl2
470 IF op$="/" THEN zahl1=zahl2 * (INT(RAN#*10)+1): ergebnis=zahl1/zahl2
480 LOCATE 0,1:PRINT"Wieviel ist";zahl1:op$:zahl2;
490 INPUT einsabe: IF einsabe=0 THEN 200
500 IF einsabe=ergebnis THEN richtig=richtig+1: PRINT"Ihre Einsabe war richtig!" ELSE
falsch=falsch+1: PRINT"Fehler! Richtig waere":ergebnis
510 GOTO 400
```

## 8.6 Übungen

### 8.6.1 Übung 8-1

Wozu dient die IF-Anweisung?

### 8.6.2 Übung 8-2

Welche Wirkung hat folgendes Programm?

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
10 INPUT EINGABE$
20 IF <EINGABE$ = "JA" THEN PRINT "DANKE"
30 IF <EINGABE$ = "NEIN" THEN PRINT "SCHADE"
40 PRINT "JA ODER NEIN": GOTO 10

```

### 8.6.3 Übung 8-3

Sind die folgenden logischen Ausdrücke zulässig?

- |                    |                       |                      |
|--------------------|-----------------------|----------------------|
| a.) $A=4$          | e.) $1>2$             | i.) NOT OR NOT TOBE  |
| b.) $B=D$ OR $C=3$ | f.) $SUMME>ZAHL$      | j.) NOT A AND NOT B  |
| c.) $A>5$          | g.) $BUCHSTABE\$="A"$ | k.) BELEAVEME OR NOT |
| d.) $5>A$          | h.) $A$ AND 1 AND 2   | l.) A NOT B          |

### 8.6.4 Übung 8-4

Ist das folgende zulässig?

- $10$  IF  $A=5$  THEN IF  $B=2$  THEN  $18$
- $20$  IF  $1$  THEN  $2$
- $30$  IF  $A$  XOR  $B$  GOTO  $A$  ELSE  $B$
- $40$  IF  $A\$+B\$=C\$$  THEN  $A\$=C\$$ : $B\$=""$ : $C\$=""$
- $50$  IF  $1<A$  THEN GOSUB  $200$  ELSE GOSUB  $300$ : GOTO  $50$
- $60$  IF  $A<5$  AND  $A>10$  THEN PRINT "Die Eingabe ist richtig!"
- $70$  IF  $A>5$  OR  $A<10$  THEN PRINT "Die Eingabe ist falsch!"

### 8.6.5 Übung 8-5

Was ist eine Programmschleife?

### 8.6.6 Übung 8-6

Schreiben Sie ein Programm, das eine Geheimzahl zwischen 1 und 100 einliest. Achten Sie darauf, dass die Zahleneingabe nicht sichtbar ist. Danach lassen Sie nach dieser Geheimzahl suchen, indem Sie immer wieder eine Zahleneingabe starten. Nach der Zahleneingabe wird einer der folgenden Hinweise ausgegeben, je nachdem was zutreffend ist:

- Ihre Eingabe ist zu klein!
- Ihre Eingabe ist zu gross!
- Sie haben die Geheimzahl gefunden!

Nachdem die Geheimzahl gefunden wurde, ist das Programm beendet.

### 8.6.7 Übung 8-7

Erweitern Sie das Programm „Rechenübung V2.01“ um folgende zwei Dinge:

- Mit der Eingabe [5] soll zufällig eine der vier Rechenarten abgefragt werden. Solange bis man mit 0 Beendet.
- Mit der Eingabe [6] soll der Schwierigkeitsgrad variiert werden von „leicht“ über „mittel“ bis „schwer“. Bisher werden die Zufallszahlen *zahl1* und

*zahl2* im Bereich von 0 bis 9. Mit dieser Änderung sollen die Zufallszahlen je nach Schwierigkeitsstufe wie folgt erstellt werden:

- Leicht 1 bis 10
- Mittel 1 bis 25
- Schwer 1 bis 100

Hier ein Vorschlag für die Menü-Gestaltung:

```
100 REM -----
110 REM Eine Rechenuebung
120 REM Version 3.00 vom 05.08.2009
130 REM Autor: Manfred Becker
140 REM -----
150 zufALL=0
160 DIM level$(3): level$(0)="leicht": level$(1)="mittel": level$(2)="schwer"
170 DIM levelM(3): levelM(0)=10: levelM(1)=25: levelM(2)=100
180 levelindex=0
200 REM Auswahlmenue
210 CLS
220 LOCATE 0,0: PRINT"[0]=Ende [1]=+ [2]=- [3]=* [4]=/";
230 LOCATE 0,1: PRINT"[5]=Zufall [6]=":level$(levelindex);
240 INPUT auswahl
```

## 9 Anweisungen wiederholen

Mit Hilfe der IF- und GOTO-Anweisungen können Sie einen Programmteil wiederholt ausführen. Der entsprechende Programmteil heißt Schleife. Die meisten Programme verwenden Schleifen. In diesem Kapitel wiederhole ich noch einmal kurz das IF / GOTO Verfahren, bevor ich Ihnen noch eine weitere Methode zeige, solche Schleifen zu bilden.

### 9.1 Das IF / GOTO-Verfahren

Das IF / GOTO-Verfahren eignet sich sehr gut bei allen Eingaben, welche überprüft werden sollen. Stellt man nach der Eingabe fest, das man sie nicht in Ordnung ist, kann mit der GOTO-Anweisung einfach wieder zurück zur Eingabestelle gesprungen werden.

Hier z.B. wird eine Passwort-eingabe ausgewertet. Stimmt die Eingabe nicht mit dem gewünschten Wert überein, so wird die Eingabe wiederholt.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINT "Passwort eingeben";
20 PW$ = INPUT$(4)
30 IF PW$<>"4711" THEN GOTO 10
40 CLS: PRINT "Passwort OK!"
```

Ein anderer typischer Fall sind Anwenderfragen, welche mit ja oder nein beantwortet werden können. Je nachdem, welche Antwort angegeben wurde, springt das Programm zu unterschiedlichen Programmstellen.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
110 PRINT "Nochmal? (j/n)";
120 CH$ = INKEY$
130 IF CH$="J" THEN GOTO 10
140 IF CH$<>"N" THEN GOTO 120
150 CLS: PRINT "Ciao!": END
```

Wenn eine Programmschleife genau N mal wiederholt werden soll, kann auch das IF / GOTO Verfahren angewendet werden.

Aber genau für solche Fälle bietet das CASIO-Basic noch eine weitere Methode – die FOR ... NEXT Schleife.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINT "Zahl eingeben";
20 INPUT N
30 S=0 : REM Summenwert auf 0
40 I=1 : REM Index auf 1
50 S=S+I
60 I=I+1: IF I<N THEN GOTO 50
70 PRINT "Die Summe der Zahlen
von 1 bis";N;" ist";S
```

### 9.2 Die FOR ... NEXT Schleife

Die FOR ... NEXT Schleife führt die Programmzeilen zwischen der FOR-Anweisung und der NEXT-Anweisung aus und erhöht die Steuervariable, beginnend mit dem Anfangswert. Die Ausführung wird beendet, wenn der Wert der Steuervariablen den spezifizierten Endwert überschreitet.

### 9.3 Summe der ersten N ganzen Zahlen

Das Summenbeispiel von eben kann mit der FOR ~ NEXT Schleife viel eleganter gelöst werden.

Der Anfang dieser FOR ~ NEXT Schleife befindet sich in Programmzeile 40. Dabei wird auch die Zählvariable I benannt, welche mit dem Wert 1 beginnt, sich mit jedem Schleifendurchlauf um eins erhöht, bis es beim Endwert N angekommen ist. Ist dieser Wert erreicht, wird die Schleife

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 PRINT "Zahl eingeben";
20 INPUT N
30 S=0 : REM Summerwert auf 0
40 FOR I=1 TO N
50 S=S+I
60 NEXT I
70 PRINT "Die Summe der Zahlen
von 1 bis";N;" ist";S
```

nicht mehr durchlaufen, und der Programmteil nach dem Schleifenende (das ist die Zeile mit dem NEXT-Befehl) wird ausgeführt.

Alle Zeilen zwischen dem FOR-Befehl und dem NEXT-Befehl gehören zu Schleife. Im obigen Beispiel ist das die Programmzeile 50, in der die Summenberechnung durchgeführt wird.

Sie sehen, dass der Wert der Zählvariable I automatisch erhöht wird. Wir benötigen somit keine zusätzliche Anweisung wie  $I=I+1$ .

Nebenbei bemerkt, muss der Name der Zählvariablen nicht zwingend I sein, obwohl I häufig dafür verwendet wird.

### 9.4 Wertetabellen

Mit Hilfe des eben gezeigten FOR ... NEXT Befehls lassen sich sehr einfach Wertetabellen berechnen und ausgeben.

Hier zeige Ich Ihnen, wie Sie mittels einer FOR ... Next Schleife die Quadrat- und Kubikzahlen für die Zahlen 1 bis 10 berechnen und ausgeben können.

Beim PRINT-Befehl wurden zur besser formatierten Ausgabe das Komma verwendet.

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 REM -----
20 REM Tabelle mit Quadrat
30 REM und Kubikzahlen fuer
40 REM die Zahlen 1 bis 10
50 REM -----
60 FOR I=1 TO 10
70 PRINT I, I^2, I^3
80 NEXT I
90 END
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
1
```

Erinnern Sie sich noch an die Wertetabelle aus Kapitel 3.1.2?

Hier sehen Sie das Programm zur Berechnung der Wertetabelle zu:

$$f(x) = 0.5x^3 - 2x^2 + 3x - 4$$

```

CAPS S CAL BASIC DEG/GRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
10 REM -----
20 REM Tabelle der Funktion
30 REM 0.5x^3 - 2x^2 + 3x-4
40 REM von 0 bis 9
50 REM -----
60 FOR x=0 TO 9
70  y = .5*x^3-2*x^2+3*x-4
80  PRINT x, y
90 NEXT I
100 END

```

## 9.5 Fortgeschrittene Schleifenstrukturen

Die FOR ... Next Schleife bietet noch zwei weitere Möglichkeiten, welche Sie bisher noch nicht verwendet haben:

- Die Zählvariable kann um jede beliebige Zahl erhöht werden. So etwas wird *variable Schrittweite* genannt.
- Eine Schleife darf innerhalb einer Schleife erzeugt werden. Solche Schleifen heißen *verschachtelte Schleifen*.

### 9.5.1 Variable Schrittweite

Mit diesem kleinen Programm werden die Zahlen 5, 10, 15, 20, usw. bis 50 ausgegeben.

```

CAPS S CAL BASIC DEG/GRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
10 REM Variable Schrittweite
20 FOR N=5 TO 50 STEP 5
30  PRINT N;
40 NEXT N

```

Sie können aber auch eine negative Schrittweite angeben. Damit ist die Ausgabe 10, 8, 6, 4, 2.

```

CAPS S CAL BASIC DEG/GRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
10 REM Variable Schrittweite
20 FOR N=10 TO 1 STEP -2
30  PRINT N;
40 NEXT N

```

Sogar dezimale Werte sind für die Schrittweite zugelassen. Die Ausgabe zu diesem Beispiel ist: 1, 1.1, 1.2, 1.3, usw. bis 2.

```

CAPS S CAL BASIC DEG/GRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
10 REM Variable Schrittweite
20 FOR N=1 TO 2 STEP 0.1
30  PRINT N;
40 NEXT N

```

## 9.5.2 Verschachtelte Schleifen

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 REM Verschachtelte Schleifen
20 FOR X=1 TO 8
30   FOR Y = 1 TO 8
40     PRINT X;Y;
50   NEXT Y
60 NEXT X
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
1
```

## 9.6 Übungen

### 9.6.1 Übung 9-1

### 9.6.2 Übung 9-2

### 9.6.3 Übung 9-3

### 9.6.4 Übung 9-4

### 9.6.5 Übung 9-5

### 9.6.6 Übung 9-6

### 9.6.7 Übung 9-7

### 9.6.8 Übung 9-8

### 9.6.9 Übung 9-9

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
x
```

## 10 Tipps für das Programmieren (Teil 2)

Die Programme, welche Sie jetzt schon schreiben können sind inzwischen schon sehr anspruchsvoll geworden. Es ist nun an der Zeit ein paar weitere Programmiertipps „an den Mann“ zu bringen, bevor wir zu den Beispielprogrammen im nächsten Kapitel kommen.

### 10.1 Weitere Grundbefehle

BEEP  
DATA, READ und RESTORE  
PEAK und POKE

### 10.2 Weitere Numerische Funktionen

SIN, COS, TAN  
LN, LOG

### 10.3 Weitere Zeichen-Funktionen

&H, HEX\$  
LEFT\$, MID\$, RIGHT\$  
ASC, CHR, STR\$  
VAL, VALF

### 10.4 Weitere Eingabe/Ausgabe-Befehle

OPEN, CLOSE, INPUT#, PRINT#,  
LLIST, LPRINT

### 10.5 Die Datenbank-Funktion nutzen

LIST#, LOAD#, NEW#, READ#, RESTORE#, WRITE#

### 10.6 Die eingebaute Wissenschaftliche Bibliothek nutzen

GOTO "LIB0:NNNN"

### 10.7 Fehlerbehandlung

ON ERROR GOTO und RESUME

### 10.8 Debugging

TRON und TROFF

## 11 Beispielprogramme

Hier finden Sie einige Beispielprogramme für Ihre Casio Pocket Computer. Sie machen deutlich, was man alles mit dem Casio BASIC machen kann.

Ich habe die Programme mit meinem Casio Notepad editiert, und zum Test auf den Pocket Computer übertragen. Mit meinem USB-Interface dauert die Übertragung nur wenige Sekunden. Dadurch konnte ich mir das etwas umständliche Editieren eines großen Programms direkt auf dem Pocket Computer ersparen (obwohl das auch machbar wäre). Jedenfalls verliert man mit der zwei Zeilen-Anzeige schnell den Überblick, wohingegen der Programmtext auf meinem Casio Notepad sogar Syntax-Highlighting unterstützt.

Wie bereits erwähnt, können Sie alle BASIC-Quelltexte von meiner Homepage herunterladen. Oder Sie kopieren das BASIC-Listing aus diesem Tutorial in die Zwischenablage und fügen es im BASIC-Editor wieder ein.

Hier eine Übersicht, der nun folgenden Beispielprogramme:

- |                        |   |
|------------------------|---|
| 1. <b>Zahlenraten</b>  | Erraten Sie eine Zufallszahl zwischen 1 und 100 |
| 2. <b>Code-Knacker</b> | Erraten Sie eine vierstellige Geheimzahl        |
| 3. <b>Hangman</b>      | Erraten Sie das Lösungswort                     |
| 4. <b>TRANSLIB</b>     | Übertragen Sie ein Library-Programm nach P0     |
| 5. <b>xxx</b>          | xxx   |
| 6. <b>xxx</b>          | xxx   |
| 7. <b>xxx</b>          | xxx   |
| 8. <b>xxx</b>          | xxx   |
| 9. <b>xxx</b>          | xxx   |
| 10. <b>xxx</b>         | xxx   |

## 11.1 Zahlenraten

## Prg. 11.1

## Das Basic-Listing zum Programm Zahlenraten

```

100 REM -----
110 REM Programm Zahlenraten
120 REM Version 1.00 vom 15.08.2009
130 REM Autor: Manfred Becker
140 REM E-Mail: mani.becker@web.de
150 REM URL: http://manib.funpic.de
160 REM -----
200 REM
210 maxzahl=100
220 maxversuche=10
230 pause=500
240 GOTO 8000 'Nochmal?
300 REM Start
310 CLS
320 LOCATE 0,0: PRINT"Ich denke mir eine Zahl aus";
330 LOCATE 0,1: PRINT"zwischen 1 und";maxzahl;
340 FORn=0TO2
350   FORi=0TO pause:NEXTi
360   PRINT".";
370 NEXTn
400 REM Zufallszahl ermitteln
410 zahl=INT(RAN#*maxzahl)+1
420 durchsams=1
500 REM Durchsams
510 CLS
520 LOCATE 0,0: PRINT"Durchsams Nr.":durchsams;
530 LOCATE 0,1: PRINT"Rate meine Zahl:";
540 INPUTa3:einsabe: LOCATE0,0: IF einsabe<=0 GOTO500
550 IF einsabe=zahl THEN GOTO 650 'Gewonnen
560 LOCATE 0,1: PRINT"Ihre Zahl":einsabe;"ist zu ";
570 IF einsabe<zahl THEN PRINT"klein!";
580 IF einsabe>zahl THEN PRINT"gross!";
590 FORi=0TO pause:NEXTi
600 durchsams=durchsams+1
610 IF durchsams>maxversuche THEN 700 'Verloren
620 GOTO 500
650 REM Gewonnen
660 CLS
670 LOCATE 0,0: PRINT"Hurra! Sie haben meine Zahl":zahl;
680 LOCATE 0,1: PRINT"in nur":durchsams;"Versuchen erraten!"
690 GOTO 8000 'Nochmal?
700 REM Verloren
710 CLS
720 LOCATE 0,0: PRINT"Oh nein! Sie haben meine Zahl":zahl;
730 LOCATE 0,1: PRINT"leider nicht erraten!"
8000 REM Nochmal?
8010 CLS
8020 LOCATE 0,0: PRINT"HANGMAN Version 1.00";
8030 LOCATE 0,1: PRINT"Wollen Sie noch einmal? (j/n)";
8040 einsabe$=INPUT$(1,0)
8050 IF einsabe$="j" OR einsabe$="J" THEN 300 'Start
8060 IF einsabe$="n" OR einsabe$="N" THEN 8080 'Ende
8070 GOTO 8040
8080 REM Ende
8090 CLS
8100 PRINT"Auf wiedersehen...";
8110 END

```

## 11.2 Code-Knacker

## Prg. 11.2 Das Basic-Listing zum Code-Knacker

```

100 REM -----
110 REM Programm CodeKnacker
120 REM Version 1.00 vom 19.08.2009
130 REM Autor: Manfred Becker
140 REM E-Mail: mani.becker@web.de
150 REM URL: http://manib.funpic.de
160 REM -----
200 REM
210 DIM zahl(4):DIM einsabe(4)
230 pause=100
240 GOTO 8000 'Nochmal?
250 REM Hinweistext
260 DATA"Programmbeschreibung", "(weiter mit [EXE])"
261 DATA"Versuchen Sie meinen 4-stelligen","seheinen Zahlencode zu knacken!"
262 DATA"Sie haben maximal 10 Versuche.", "Keine Ziffer kommt doppelt vor!"
263 DATA"Nach der Einsabe erfolgt die", "Auswertung mittels zwei Nummern."
264 DATA"Die erste Nummer zeist die", "Treffer an der richtigen Stelle."
265 DATA"Die zweite Nummer zeist die", "Treffer an einer anderen Stelle."
266 DATA"Gewonnen haben Sie, wenn alle", "Ziffern ueberein stimmen."
300 REM Start
310 RESTORE260
320 FORn=0 TO 6
330 READ zeile1$,zeile2$
340 CLS
350 PRINTzeile1$:
360 LOCATE 0,1: PRINTzeile2$:
370 LOCATE 0,0: PRINT""
380 NEXTn
400 REM Zufallszahl ermitteln
410 CLS:zahl$=""
420 LOCATE 0,0: PRINT"Ich generiere den Zahlencode":
430 LOCATE 0,1: PRINT"Code: ";
440 FORn=0TO3
450 zahl(n)=INT(RAN#*10)
460 FORm=0TO n-1
470 IF zahl(m)=zahl(n) THEN 450 'andere Zufallszahl
480 NEXTm
490 PRINT"*";
500 zahl$=zahl$+MID$(STR$(zahl(n)),2,1)
510 FORp=0TOpause:NEXTp
520 NEXTn
530 PRINT" OK! Weiter mit [EXE]"
550 CLS
560 x=0:y=0:durchsams=1
600 REM Zifferneinsabe
610 LOCATEx,y:PRINT"?????";
620 LOCATEx,y:INPUT@4:einsabe$:LOCATE0,0
630 REM Einsabe ueberpruefen...
640 IF LEN(einsabe$)<>4 THEN 600 'Einsabe wiederholen
650 FORn=0TO3
660 ch$=MID(einsabe$,n+1,1) 'Ziffer auslesen
670 IF ch$<"0" OR ch$>"9" THEN 600 'Einsabe wiederholen
680 einsabe(n)=VAL(ch$) 'Zahl zuweisen
690 FORm=0TO n-1
700 IF einsabe(m)=einsabe(n) THEN 600 'Einsabe wiederholen
710 NEXTm
720 NEXTn
800 REM Einsabe auswerten
810 ziffer1=144:ziffer2=144
820 FORn=0TO3
830 IF einsabe(n)=zahl(n) THEN ziffer1=ziffer1+1
840 FORm=0TO3
850 IF n<>m AND einsabe(n)=zahl(m) THEN ziffer2=ziffer2+1
860 NEXTm
870 NEXTn
880 LOCATEx+4,y:PRINT CHR$(ziffer1):CHR$(ziffer2):
890 IF ziffer1=148 AND ziffer2=144 THEN 1000 'Gewonnen!

```

## Prg. 11.2 Das Basic-Listing zum Code-Knacker

```
900 REM Naechste Eingabeposition
910 x=x+6:IF x=30 THEN x=0:y=y+1:IF y=2 THEN 1100 'Verloren
920 durchsaans=durchsaans+1
930 GOTO 600 'Eingabe wiederholen
1000 REM Gewonnen
1010 CLS
1020 LOCATE 0,0: PRINT"Hurra! Sie haben meine Zahl ";zahl$;
1030 LOCATE 0,1: PRINT"mit nur";durchsaans;"Versuchen erraten!"
1040 GOTO 8000 'Nochmal?
1100 REM Verloren
1110 CLS
1120 LOCATE 0,0: PRINT"Oh nein! Sie haben meine Zahl";
1130 LOCATE 0,1: PRINTzahl$;" leider nicht erraten!"
8000 REM Nochmal?
8010 CLS
8020 LOCATE 0,0: PRINT"CodeKnacker Version 1.00";
8030 LOCATE 0,1: PRINT"Wollen Sie noch einmal? (j/n)";
8040 einsabe$=INPUT$(1,0)
8050 IF einsabe$="j" OR einsabe$="J" THEN 300 'Start
8060 IF einsabe$="n" OR einsabe$="N" THEN 8080 'Ende
8070 GOTO 8040
8080 REM Ende
8090 CLS
8100 PRINT"Auf wiedersehen...";
8110 END
```

## 11.3 Hangman

## Prg. 11.3 Das Basic-Listing zum Hangman Spiel

```

100 REM -----
110 REM Programm Hangman
120 REM Version 1.00 vom 18.08.2009
130 REM Autor: Manfred Becker
140 REM E-Mail: mani.becker@web.de
150 REM URL: http://manib.funpic.de
160 REM -----
200 CLEAR: GOTO 8000 'Nochmal?
300 REM Textvorschabe
310 CLS: INPUT "Bitte geben Sie Ihren Text ein: "; I$
320 CLS: A$="": B$="": FOR I=1 TO 31: B$=B$+CHR$(160): NEXT I
330 PAUSE=250
340 VERSUCHE=9
350 FOR I=1 TO LEN(I$)
360 IF MID$(I$, I, 1) = " " THEN A$=A$+" " ELSE A$=A$+"-"
370 NEXT I
380 REM Zeichen-Eingabe
390 LOCATE 0, 0: PRINT A$:
400 LOCATE 0, 1: PRINT "Machen Sie nun Ihre Eingabe ";
410 FOR I=0 TO PAUSE: NEXT I
420 LOCATE 0, 1: PRINT B$:
430 C$=INKEY$: IFC$="" OR C$=" " THEN 430
435 LOCATE 0, 1: PRINT "Pruefe Ihre Eingabe: "; C$:
440 FOUND=0: FOR I=1 TO LEN(B$)
450 IF MID$(B$, I, 1)=C$ THEN FOUND=1
460 NEXT I
470 IF FOUND=0 THEN 530
480 REM Doppelte Zeichen-Eingabe
490 LOCATE 0, 1: PRINT "Buchstabe "+C$+" gab's schon mal!";
500 FOR I=0 TO PAUSE: NEXT I
510 GOTO 390
520 REM Ueberpruefung
530 FOUND=0: FOR I=1 TO LEN(I$)
540 IF MID$(I$, I, 1)=C$ THEN FOUND=1
550 NEXT I
560 I=0
570 I=I+1: IF (MID$(B$, I, 1) < C$) THEN 570
580 B$=LEFT$(B$, I-1)+C$+MID$(B$, I, 31-I)
590 IF FOUND=1 THEN 700
600 REM Eingabe nicht vorhanden
610 VERSUCHE=VERSUCHE-1
620 LOCATE 0, 1: PRINT "Buchstabe "+C$+" kommt nicht vor.";
630 FOR I=0 TO PAUSE: NEXT I
640 IF VERSUCHE=0 THEN 900
650 LOCATE 0, 1: PRINT "Sie haben noch"; VERSUCHE; " Versuche.";
660 FOR I=0 TO PAUSE: NEXT I
670 GOTO 390
700 REM Eingabe vorhanden
710 Z=0: E=1: FOR I=1 TO LEN(I$)
720 IF MID$(I$, I, 1)=C$ THEN Z=Z+1: A$=LEFT$(A$, I-1)+C$+MID$(A$, I, LEN(A$)-I)
730 IF MID$(A$, I, 1)="-" THEN E=0
740 NEXT I
750 LOCATE 0, 1: PRINT C$; " kommt"; Z; "mal vor. ";
760 FOR I=0 TO 2*PAUSE: NEXT I
770 IF E=0 THEN 390
800 REM Gewonnen...
810 LOCATE 0, 0: PRINT I$:
820 FORM=1 TO 5
830 LOCATE 0, 1: PRINT " ";
840 FOR I=0 TO PAUSE: NEXT I
850 LOCATE 0, 1: PRINT "HURRA!!! Sie haben gewonnen! ";
860 FOR I=0 TO PAUSE: NEXT I
870 NEXT M
880 GOTO 8000
900 REM Verloren...
910 LOCATE 0, 1: PRINT "Sie haben leider verloren! ";
920 FOR I=0 TO 2*PAUSE: NEXT I

```

## Prg. 11.3 Das Basic-Listing zum Hangman Spiel

```
930 LOCATE 0,1:PRINT"Das war die richtige Loesung! ";
940 FORI=0TO2*PAUSE:NEXTI
950 LOCATE 0,0:PRINTI#;
960 FORI=0TO5*PAUSE:NEXTI
8000 REM Nochmal?
8010 CLS
8020 LOCATE 0,0: PRINT"HANGMAN Version 1.00";
8030 LOCATE 0,1: PRINT"Wollen Sie noch einmal? (j/n)";
8040 einsabe$=INPUT$(1,0)
8050 IF einsabe$="j" OR einsabe$="J" THEN 300 'Start
8060 IF einsabe$="n" OR einsabe$="N" THEN 8080 'Ende
8070 GOTO 8040
8080 REM Ende
8090 CLS
8100 PRINT"Auf wiedersehen...";
8110 END
```

## 11.4 TRANSLIB

Dieses Programm überträgt ein beliebiges Casio Library-Programm in den Programmbereich P0 ihres Casio FX-850P/FX-880P.

Stellen Sie vor dem Übertragen sicher, dass genügend Speicherplatz auf Ihrem Casio-Rechner vorhanden ist (Der Basic-Befehl NEW löscht ein Programm im aktuellen Programmbereich. Falls ihr alle Programme löschen wollen, verwendet den Basic-Befehl NEW ALL. Denkt aber vorher an Ihre Datensicherung!

Nun kopieren Sie das Programm TRANSLIB1 nach P1 und das Programm TRANSL2 nach P2. Da das Library-Programm in P0 erzeugt wird, sollten Sie diesen Programmbereich unbedingt löschen (falls noch nicht geschehen). Starten Sie das Programm im Programmbereich P1, und geben Sie die gewünschte LIBRARY-Nummer ein. Sobald Sie eine Nummer eingegeben haben, wird das Library- Programm im Programmbereich P0 erstellt. Dort können Sie es listen, starten oder abspeichern (aber nicht editieren!). Das Programm im Programmbereich P2 macht den Programmbereich P0 wieder normal nutzbar. Jetzt kann das Programm auf einen PC übertragen werden.

### Prg. 11.4 TRANLIB 1

```

10 REM Programm:TRANS-LIB -> P1
11 REM Aufgabe :Hauptprogramm
20 CLEAR: CLS: PRINT "Dieses Hilfsprogramm uebertraest ein Library-Pros. nach P0."
30 INPUT "Bitte geben Sie die Library- Programm-Nr. ein :":NR$
40 DEFSEG=0: X=PEEK1715+PEEK1716*256-15
50 A=PEEK1718+PEEK1719*256+PEEK1720*65536
60 B=PEEK1724+PEEK1725*256+PEEK1726*65536
70 FOR I=A TO B-15 STEP 15
80 DEFSEG=I/16: C=I-INT(I/16)*16+6
90 FOR K=1 TO LEN(NR$): IF ASC(MID$(NR$,K)=PEEK(K+C) THEN NEXT K: GOTO 110
100 NEXT I: PRINT "Gesuchtes Library-Programm nicht gefunden. Bitte neu waehlen!"
: GOTO 10
110 REM Adressen ermitteln
120 DEFSEG=0: IF PEEK(X+2)=0 THEN A0=PEEK X: A1=PEEK(X+1): A3=PEEK(X+3)
: A4=PEEK(X+4)
130 FOR K=0 TO 5: DEFSEG=I/16: N=PEEK(K+I-INT(I/16)*16): DEFSEG=0: POKE X+K,N
: NEXT K
140 PRINT "Library-Programm-Nr.":NR$: PRINT " ist nun in P0 gespeichert!"

```

### Prg. 11.4b TRANLIB 2

```

10 REM Programm:TRANS-LIB -> P2
11 REM Aufgabe :P0 loeschen
20 INPUT "P0 loeschen? [J/N] ":Q$
30 IF Q$<>"J" OR X=0 THEN END
40 POKE X+0,A0
41 POKE X+1,A1
42 POKE X+2,0
43 POKE X+3,A3
44 POKE X+4,A4
45 POKE X+5,0
46 X=0

```

**11.5xxxx**

<b>Prg. 11.5</b>	XXXX
------------------	------

**11.6xxxx**

<b>Prg. 11.6</b>	XXXX
------------------	------

**11.7xxxx**

<b>Prg. 11.7</b>	XXXX
------------------	------

**11.8xxxx**

<b>Prg. 11.8</b>	XXXX
------------------	------

**11.9xxxx**

<b>Prg. 11.9</b>	XXXX
------------------	------

**11.10xxxx**

<b>Prg. 11.10</b>	XXXX
-------------------	------

**Prg. 11.10** XXXX

## 12 Anhang

### 12.1 Antworten zu den Übungen

#### 12.1.1 Antworten zu Übungen aus Kapitel 5

Übung 5-1:	PRINT (5+6) / (1 + 2/3)	6.6
Übung 5-2:	PRINT 1 + (1/2) * (1 / (1 + (1/2))) oder PRINT 1 + .5 * (1 / (1 + .5))	1.333333333
Übung 5-3:	PRINT 20 * 9/5 + 32	68
Übung 5-4:	PRINT 40000 / 24	1666.666667
Übung 5-5:	PRINT 24*60*60 PRINT 24*60*60*7 PRINT 24*60*60*30 PRINT 24*60*60*365	86400 604800 2592000 30758400
Übung 5-6:	PRINT 350 / 55	6.363636364
Übung 5-7:	PRINT SQR(3^2 + 4^2)	5
Übung 5-8:	PRINT SQR(3^2 * 4^2 / 5^2)	2.4
Übung 5-9:	PRINT "Die Summe von";A;" +";B;" =";A+B	

#### 12.1.2 Antworten zu Übungen aus Kapitel 6

##### Übung 6-1:

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
10 REM -----
20 REM Programm zu Uebung 6-1
30 REM -----
40 PRINT "Bitte 4 Zahlen einseben... "
50 INPUT A,B,C,D
60 SUMME = A+B+C+D
70 PRINT "Die Summe der vier Zahlen ist";SUMME
80 PRINT "Der Mittelwert der vier Zahlen ist";SUMME/4
90 PRINT "Das Produkt der vier Zahlen ist";A*B*C*D

```

Übung 6-2:	a.) nein	e.) ja	i.) ja
	b.) ja	f.) ja	j.) nein
	c.) nein	g.) nein	k.) ja
	d.) ja	h.) nein	l.) ja

##### Übung 6-3:

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFN PRT TR STOP
10 REM -----
20 REM Programm zu Uebung 6-3
30 REM -----
40 PRINT "Bitte geben Sie Ihren Namen ein:"
50 INPUT NAME$
60 PRINT "Hallo ";NAME$;"! Ihr Name hat";LEN(NAME$);"
Buchstaben."

```

## Übung 6-4:

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 REM -----
20 REM Programm zu Uebung 6-4
30 REM -----
40 PRINT "Bitte geben Sie die drei Seiten eines
rechtwinkligen Dreiecks ein:"
50 INPUT A,B,C
60 IF A=0 AND B>0 AND C>0 THEN PRINT "A =";SQR(C*B*B)
70 IF A>0 AND B=0 AND C>0 THEN PRINT "B =";SQR(C*A*A)
80 IF A>0 AND B>0 AND C=0 THEN PRINT "C =";SQR(A*A+B*B)

```

## Übung 6-5:

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 REM -----
20 REM Programm zu Uebung 6-5
30 REM -----
40 PRINT "Bitte geben Sie den Radius eines Kreises ein:"
50 INPUT r
60 PRINT "Durchmesser =";2*r
70 PRINT "Umfang =";2*PI*r
80 PRINT "Flaeche =";PI*r^2

```

- Übung 6-6:
- |          |          |          |
|----------|----------|----------|
| a.) nein | e.) nein | i.) nein |
| b.) ja   | f.) ja   | j.) nein |
| c.) ja   | g.) ja   | k.) nein |
| d.) nein | h.) nein | l.) nein |

## Übung 6-7:

```

CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
10 REM -----
20 REM Programm zu Uebung 6-7
30 REM -----
40 PRINT "Bitte geben Sie die Seitenlaenge eines
Wuerfels ein:"
50 INPUT a
60 PRINT "Volumen =";a^3
70 PRINT "Oberflaeche =";6*a^2
80 PRINT "Raumdiagonale =";a*SQR(3)

```

## 12.1.3 Antworten zu Übungen aus Kapitel 8

## Listing 6 Rechenübung.300.cas

(37 Programmzeilen)

```

100 REM -----
110 REM Eine Rechenuebung
120 REM Version 3.00 vom 05.08.2009
130 REM Autor: Manfred Becker
140 REM -----
150 zufALL=0
160 DIM level$(3): level$(0)="leicht": level$(1)="mittel": level$(2)="schwer"
170 DIM levelM(3): levelM(0)=10: levelM(1)=25: levelM(2)=100
180 levelindex=0
200 REM Auswahlmenue
210 CLS
220 LOCATE 0,0: PRINT"[0]=Ende [1]=+ [2]=- [3]=* [4]=/";
230 LOCATE 0,1: PRINT"[5]=Zufall [6]=":level$(levelindex);
240 INPUT auswahl
250 IF auswahl<0 OR auswahl>6 THEN PRINT"Falsche Eingabe!": GOTO 200
260 IF auswahl=0 THEN PRINT"Auf wiedersehen!": END
262 IF auswahl=5 THEN zufALL=-1 ELSE zufALL=0
264 IF auswahl=6 THEN levelindex=levelindex+1: IF levelindex=3 THEN levelindex=0
266 IF auswahl=6 THEN 200
270 richtig=0: falsch=0: max = levelM(levelindex)
275 IF zufALL THEN auswahl=INT(RAN#*4)+1
280 IF auswahl=1 THEN op$="+": titel$="Addition"
290 IF auswahl=2 THEN op$="-": titel$="Subtraktion"

```

## Listing 6

Rechenübung.300.cas

(37 Programmzeilen)

```
300 IF auswahl=3 THEN op$="*": titel$="Multiplikation"
310 IF auswahl=4 THEN op$="/": titel$="Division"
400 REM Berechnung
410 CLS
420 LOCATE 0,0: PRINT titel$;" ":richtig:"richtig ";falsch:"falsch";
430 zahl2=INT(RAN#*max)+1
440 IF op$="+" THEN zahl1=INT(RAN#*max)+1: ergebnis=zahl1+zahl2
450 IF op$="-" THEN zahl1=zahl2 + (INT(RAN#*max)+1): ergebnis=zahl1-zahl2
460 IF op$="*" THEN zahl1=INT(RAN#*max)+1: ergebnis=zahl1*zahl2
470 IF op$="/" THEN zahl1=zahl2 * (INT(RAN#*max)+1): ergebnis=zahl1/zahl2
480 LOCATE 0,1:PRINT"Wieviel ist";zahl1;op$:zahl2;
490 INPUT einsabe: IF einsabe=0 THEN 200
500 IF einsabe=ergebnis THEN richtig=richtig+1: PRINT"Ihre Einsabe war richtig!" ELSE
falsch=falsch+1: PRINT"Fehler! Richtig waere";ergebnis
510 IF zufall GOTO 275: ELSE GOTO 400
```

## 12.1.4 Antworten zu Übungen aus Kapitel 9

## 12.2 Begriffserklärung

Hier werden ein paar Begriffe erklärt, welche in den vorhergehenden Kapiteln des Öfteren verwendet werden.

### 12.2.1 Interpreter

Als Interpreter wird ein Programm bezeichnet, das Source-Code Stück für Stück (meistens Zeile für Zeile) in ausführbaren Maschinencode übersetzt und auch sofort ausführt. Aus diesem Grund sind Interpretersprachen auch in der Ausführung langsamer als Sprachen, die einen Compiler benutzen, da die Übersetzung sich direkt auf die Laufzeit niederschlägt. Zur Optimierung gibt es verschiedenste Strategien bei der Übersetzung zur Laufzeit, aber damit wollen wir uns hier nicht näher beschäftigen.

### 12.2.2 Compiler

Als Compiler wird ein Übersetzer bezeichnet, der Source-Code in Object-Code (siehe unten) transferiert. Im üblichen Fall wird je ein File mit Source-Code vom Compiler in je ein File mit Object-Code übersetzt. Dementsprechend können bei einem Compilerlauf viele Object-Files entstehen.

### 12.2.3 Assembler

Im Prinzip ist die Aufgabe eines Assemblers dieselbe wie die eines Compilers. Auch er übersetzt Source-Code in Object-Code. Der einzige Unterschied zwischen einem Compiler und einem Assembler ist, dass ein Compiler Source-Code aus höheren Programmiersprachen übersetzt. Ein Assembler übersetzt Source-Code, der in Maschinensprache (siehe niedrige Programmiersprachen) geschrieben ist.

### 12.2.4 Object-Code

Der Begriff Object-Code bezeichnet Maschinencode, der noch nicht direkt ausführbar ist, da er teilweise noch aus dem Source-Code übernommene symbolische Namen (z.B. Funktionsaufrufe) enthält. Der Computer versteht aber intern keine symbolischen Namen, sondern nur Adressen. Daher stellt der Object-Code auch nur ein Zwischenprodukt dar, das erst in einem weiteren Verarbeitungsschritt (durch den Linker) zu einem Executable wird. Object-Code entsteht als Resultat der Übersetzung von Source-Code durch einen Compiler oder Assembler.

### 12.2.5 Library

Als Library wird eine Zusammenfassung von nützlichen Funktionen, Prozeduren etc. bezeichnet, die in verschiedenen Programmen verwendet werden können. Im Prinzip liegt eine Library selbst als Object-Code vor. Und natürlich ist auch hier wieder der Linker verantwortlich, dass eine verwendete Library zum Executable dazugebunden wird.

Prinzipiell gibt es statische und dynamische Libraries. Statische Libraries werden zum Zeitpunkt des Linkens in das ausführbare Programm übernommen, machen dieses also dementsprechend größer. Dynamische Libraries werden nicht direkt

zum Programm gelinkt, sondern es werden nur die entsprechenden Einstiegspunkte im Programm vermerkt. Dynamische Libraries müssen immer zur Laufzeit eines Programms am entsprechenden Zielrechner vorhanden sein, da es sonst nicht ausführbar ist.

### 12.2.6 Linker

Der Linker ist dafür verantwortlich, ein oder mehrere Files mit Object-Code und benötigte Libraries zu einem tatsächlich am Computer ausführbaren Programm zu binden. Hierbei muss er die noch im Object-Code vorhandenen symbolischen Namen auflösen und durch entsprechende Adressen ersetzen (=Referenzauflösung). Auch muss der Linker die korrekten Einstiegspunkte bei Verwendung von dynamischen Libraries erzeugen.

### 12.3 Befehls-Übersicht

Das ist meine alphabetisch geordnete Befehls-Übersicht. Die Seitenzahl verweist direkt zur Seite der CASIO-Bedienungsanleitung des Rechners.

Befehl/Funktion	Kategorie	Beschreibung	Seite
<b>&amp;H</b>	Zeichen-Funktion	Wandelt den auf &H folgenden 1- bis 4-stelligen Hexadezimalwert in einen Dezimalwert um.	159
<b>ABS</b>	Numerische Funktion	Berechnet den Absolutwert des Arguments.	141
<b>ACS</b>	Numerische Funktion	Berechnet den Wert des entsprechenden inversen trigonometrischen Funktionswertes für das Argument.	135
<b>ANGLE</b>	Numerische Funktion	Spezifiziert die Winkleinheit.	133
<b>ASC</b>	Zeichen-Funktion	Liefert den Zeichencode, der dem ersten Zeichen (ganz links) einer Zeichenfolge entspricht.	151
<b>ASN</b>	Numerische Funktion	Berechnet den Wert des entsprechenden inversen trigonometrischen Funktionswertes für das Argument.	135
<b>ATN</b>	Numerische Funktion	Berechnet den Wert des entsprechenden inversen trigonometrischen Funktionswertes für das Argument.	135
<b>BEEP</b>	Grundbefehl	Aktiviert den Summer.	121
<b>CHR\$</b>	Zeichen-Funktion	Liefert ein Zeichen, das dem spezifizierten Zeichencode entspricht.	150
<b>CLEAR</b>	Manueller Befehl	Löscht alle Variable und bestimmt die Variablenbereich-Größe entsprechend dem eingegebenen Parameter. Außerdem werden alle offenen Daten geschlossen.	93
<b>CLOSE</b>	Ein/Ausgabe-Befehl	Schließt Dateien und deklariert das Ende der Verwendung des I/O (Eingabe/Ausgabe)-Puffers.	166
<b>CLS</b>	Grundbefehl	Löscht den Bildschirm.	119
<b>COS</b>	Numerische Funktion	Berechnet den Wert des entsprechenden trigonometrischen Funktionswertes für das Argument.	134
<b>CUR</b>	Numerische Funktion	Berechnet die Kubikwurzel des Arguments.	140
<b>DATA</b>	Grundbefehl	Nimmt Daten auf, die durch die Anweisung READ gelesen werden sollen.	112
<b>DEFSEG</b>	Grundbefehl	Spezifiziert Segment-Grundadressen.	129
<b>DEG</b>	Zeichen-Funktion	Wandelt einen Sexagesimalwert in einen Dezimalwert um.	160
<b>DIM</b>	Grundbefehl	Deklariert ein Feld.	125
<b>DMS\$</b>	Zeichen-Funktion	Wandelt einen Dezimalwert in eine Sexagesimal-Zeichenfolge um.	161
<b>EDIT</b>	Manueller Befehl	Aktiviert den BASIC-Editor-Modus.	97
<b>END</b>	Grundbefehl	Beendet die Programmausführung.	100
<b>EOF</b>	Ein/Ausgabe-Befehl	Zeigt das Ende der Datei-Eingabe an.	170
<b>ERASE</b>	Grundbefehl	Löscht ein spezifiziertes Feld.	126
<b>ERL</b>	Grundbefehl	Liefert die Nummer einer Zeile, in der ein Fehler erzeugt wurde.	132
<b>ERR</b>	Grundbefehl	Liefert den Fehlercode, der einem aufgetretenen Fehler entspricht.	132

Befehl/Funktion	Kategorie	Beschreibung	Seite
<b>EXP</b>	Numerische Funktion	Berechnet den Wert der Exponentialfunktion für das Argument.	138
<b>FACT</b>	Numerische Funktion	Berechnet die Fakultät des Arguments.	146
<b>FIX</b>	Numerische Funktion	Ermittelt den ganzzahligen Anteil des Arguments.	143
<b>FOR ~ NEXT</b>	Grundbefehl	Führt die Programmzeilen zwischen der FOR-Anweisung und der NEXT-Anweisung aus und erhöht die Steuervariable, beginnend mit dem Anfangswert. Die Ausführung wird beendet, wenn der Wert der Steuervariablen den spezifizierten Endwert überschreitet.	108
<b>FRAC</b>	Numerische Funktion	Ermittelt die Nachkommastellen des Arguments.	144
<b>FRE</b>	Manueller Befehl	Liefert die Speicherbereich-Größe entsprechend dem Argument.	95
<b>GOSUB</b>	Grundbefehl	Springt zu einem spezifizierten Unterprogramm.	103
<b>GOTO</b>	Grundbefehl	Unbedingte Verzweigung zum spezifizierten Verzweigungsziel.	102
<b>HEX\$</b>	Zeichen-Funktion	Liefert eine Hexadezimal-Zeichenfolge für einen im Argument spezifizierten Dezimalwert.	158
<b>HYP ACS</b>	Numerische Funktion	Berechnet den Wert des entsprechenden inversen Hyperbelfunktion für das Argument.	137
<b>HYP ASN</b>	Numerische Funktion	Berechnet den Wert des entsprechenden inversen Hyperbelfunktion für das Argument.	137
<b>HYP ATN</b>	Numerische Funktion	Berechnet den Wert des entsprechenden inversen Hyperbelfunktion für das Argument.	137
<b>HYP COS</b>	Numerische Funktion	Berechnet den Wert des entsprechenden Hyperbelfunktion für das Argument.	136
<b>HYP SIN</b>	Numerische Funktion	Berechnet den Wert des entsprechenden Hyperbelfunktion für das Argument.	136
<b>HYP TAN</b>	Numerische Funktion	Berechnet den Wert des entsprechenden Hyperbelfunktion für das Argument.	136
<b>IF ~ THEN ~ ELSE IF ~ GOTO ~ ELSE</b>	Grundbefehl	Führt die THEN-Anweisung oder GOTO-Anweisung aus, wenn die spezifizierte Bedingung erfüllt wird. Die ELSE-Anweisung wird ausgeführt, wenn die spezifizierte Bedingung nicht erfüllt wird.	107
<b>INKEY\$</b>	Grundbefehl	Weist eine einzelne Zeicheneingabe von der Tastatur einer Variablen zu.	123
<b>INPUT</b>	Grundbefehl	Weist Daten, die über die Tastatur eingegeben werden, Variablen zu.	122
<b>INPUT#</b>	Ein/Ausgabe-Befehl	Liest Daten aus einer sequentiellen Datei.	168
<b>INPUT\$</b>	Grundbefehl	Weist eine spezifizierte Anzahl Zeichen von der Tastatur einer Variablen zu.	124
<b>INPUT\$</b>	Ein/Ausgabe-Befehl	Liest die spezifizierte Anzahl Zeichen aus einer sequentiellen Datei.	169
<b>INT</b>	Numerische Funktion	Ermittelt die größte ganze Zahl, die den Wert des Arguments nicht überschreitet.	143
<b>LEFT\$</b>	Zeichen-Funktion	Liefert eine spezifizierte Anzahl Zeichen vom linken Ende einer Zeichenfolge.	157
<b>LEN</b>	Zeichen-Funktion	Liefert einen Wert, der die Anzahl der Zeichen einer Zeichenfolge angibt.	158
<b>LET</b>	Grundbefehl	Weist den Wert eines Ausdrucks an der rechten Seite einer Gleichung der Variablen an der linken Seite zu.	111

Befehl/Funktion	Kategorie	Beschreibung	Seite
<b>LIST</b> <b>LIST ALL</b>	Manueller Befehl	Zeigt das gesamte oder einen Teil des gegenwärtigen spezifizierten Programms an.	96
<b>LIST#</b>	Datenbank-Befehl	Zeigt alle Datenbank-Daten an.	174
<b>LLIST</b>	Ein/Ausgabe-Befehl	Ausgabe von Programminhalten an den Drucker.	162
<b>LLIST#</b>	Datenbank-Befehl	Gibt alle Datenbank-Daten auf dem Drucker aus.	175
<b>LN</b>	Numerische Funktion	Berechnet den Wert der entsprechenden Logarithmusfunktion für das Argument	139
<b>LOAD</b> <b>LOAD ALL</b>	Ein/Ausgabe-Befehl	Liest aus einer Datei in den Speicher.	172
<b>LOAD#</b>	Datenbank-Befehl	Liest Daten in den Datenbank-Bereich.	176
<b>LOCATE</b>	Grundbefehl	Bewegt den Cursor zu einer spezifizierten Position auf dem virtuellen Bildschirm.	118
<b>LOG</b>	Numerische Funktion	Berechnet den Wert der entsprechenden Logarithmusfunktion für das Argument	139
<b>LPRINT</b>	Ein/Ausgabe-Befehl	Ausgabe von text an den Drucker.	163
<b>MID\$</b>	Zeichen-Funktion	Liefert eine spezifizierte Anzahl Zeichen von einer spezifizierten Position in einer Zeichenfolge.	155
<b>NCR</b>	Numerische Funktion	Berechnet die Kombination $nCr$ für die Werte von $n$ und $r$ .	147
<b>NEW</b> <b>NEW ALL</b>	Manueller Befehl	Löscht ein Programm.	93
<b>NEW#</b>	Datenbank-Befehl	Löscht Datenbank-Daten.	174
<b>NPR</b>	Numerische Funktion	Berechnet die Permutation $nPr$ für die Werte von $n$ und $r$ .	147
<b>ON ERROR</b> <b>GOTO</b>	Grundbefehl	Spezifiziert die Zeilennummer, zu der die Ausführung verzweigt wird, wenn ein Fehler auftritt.	130
<b>ON GOSUB</b>	Grundbefehl	Sprung zum spezifizierten Unterprogramm entsprechend der spezifizierten Verzweigungsbedingung.	106
<b>ON GOTO</b>	Grundbefehl	Springt zum spezifizierten Verzweigungsziel entsprechend der spezifizierten Verzweigungsbedingung.	105
<b>OPEN</b>	Ein/Ausgabe-Befehl	Eröffnet eine Datei zur Verarbeitung.	165
<b>PASS</b>	Manueller Befehl	Spezifiziert oder löscht ein Passwort.	92
<b>PEEK</b>	Grundbefehl	Liefert den Wert, der in der spezifizierten Speicheradresse gespeichert ist.	127
<b>PI</b>	Numerische Funktion	Gibt den Wert von $\pi$ an.	146
<b>POKE</b>	Grundbefehl	Schreibt Daten in eine spezifizierte Adresse.	128
<b>POL</b>	Numerische Funktion	Wandelt kartesische Koordinaten $(x, Y)$ in Polkoordinaten $(r, \theta)$ um.	148
<b>PRINT</b>	Grundbefehl	Zeigt Daten auf dem Bildschirm an.	115
<b>PRINT#</b>	Ein/Ausgabe-Befehl	Ausgabe von Daten in eine sequentielle Datei.	167
<b>RAN#</b>	Numerische Funktion	Erzeugt einen Zufallswert im Bereich von 0 bis 1.	145
<b>READ</b>	Grundbefehl	Liest den Inhalt der DATA-Anweisung in den Speicher.	113
<b>READ#</b>	Datenbank-Befehl	Liest Daten aus dem Datenabank-Bereich.	177
<b>REC</b>	Numerische Funktion	Wandelt Polkoordinaten $(r, \theta)$ in kartesische Koordinaten $(x, Y)$ um.	149

Befehl/Funktion	Kategorie	Beschreibung	Seite
<b>REM ( ' )</b>	Grundbefehl	Ermöglicht das Einfügen von Anmerkungen oder Kommentaren in ein Programm. Dieser Befehl wird nicht ausgeführt.	110
<b>RESTORE</b>	Grundbefehl	Spezifiziert eine DATA-Zeile zum Lesen durch die READ-Anweisung.	114
<b>RESTORE#</b>	Datenbank-Befehl	Sucht bestimmte Daten im Datenbank-Bereich und verändert die Lesereihenfolge von Datenbank-Daten.	178
<b>RESUME</b>	Grundbefehl	Bewirkt Rückkehr aus einer Fehlerabwicklungsroutine zum Hauptprogramm.	131
<b>RETURN</b>	Grundbefehl	Rückkehr der Ausführung von einem Unterprogramm zum Hauptprogramm.	104
<b>RIGHT\$</b>	Zeichen-Funktion	Liefert eine spezifizierte Anzahl Zeichen von der rechten Seite einer Zeichenfolge.	156
<b>ROUND</b>	Numerische Funktion	Rundet das Argument an der spezifizierten Stelle.	144
<b>RUN</b>	Manueller Befehl	Führt ein Programm aus.	98
<b>SAVE SAVE ALL</b>	Ein/Ausgabe-Befehl	Sichert ein Programm in eine spezifizierten Datei.	171
<b>SAVE#</b>	Datenbank-Befehl	Gibt Datenbank-Daten zu einer Datei aus, die durch den Dateideskriptor spezifiziert wurde.	175
<b>SET</b>	Grundbefehl	Spezifiziert das Ausgabeformat von numerischen Daten.	120
<b>SGN</b>	Numerische Funktion	Ermittelt den Wert, der dem Vorzeichen des Arguments entspricht.	142
<b>SIN</b>	Numerische Funktion	Berechnet den Wert des entsprechenden trigonometrischen Funktionswertes für das Argument.	134
<b>SQR</b>	Numerische Funktion	Berechnet die Quadratwurzel des Arguments.	140
<b>STOP</b>	Grundbefehl	Unterbricht die Programmausführung.	101
<b>STR\$</b>	Zeichen-Funktion	Wandelt das Argument (numerischer Wert oder Wert eines numerischen Ausdrucks) in eine Zeichenfolge um.	152
<b>TAB</b>	Grundbefehl	Gibt eine Horizontal-Tabulatorspezifikation zum Bildschirm oder Drucker aus.	117
<b>TAN</b>	Numerische Funktion	Berechnet den Wert des entsprechenden trigonometrischen Funktionswertes für das Argument.	134
<b>TROFF</b>	Manueller Befehl	Hebt den Ablaufverfolgungs-Modus auf.	99
<b>TRON</b>	Manueller Befehl	Spezifiziert den Ablaufverfolgungs-Modus.	99
<b>VAL</b>	Zeichen-Funktion	Wandelt eine numerische Zeichenfolge in einen numerischen Wert um.	153
<b>VALF</b>	Zeichen-Funktion	Führt mit einem numerischen Ausdruck, der als Zeichenfolge ausgedrückt ist, eine Berechnung aus und liefert das Ergebnis.	154
<b>VARLIST</b>	Manueller Befehl	Zeigt Variablennamen und Feldnamen an.	97
<b>VERIFY</b>	Ein/Ausgabe-Befehl	Überprüft den Inhalt einer auf Kassette gespeicherten Datei.	173
<b>WRITE#</b>	Datenbank-Befehl	Überschreibt und löscht Datenbank-Daten.	179

## 12.4 Übersicht der Wissenschaftlichen Bibliothek

Das ist eine Übersicht der Wissenschaftlichen Bibliothek. Die Seitenzahl verweist direkt zur Seite der CASIO-Bedienungsanleitung des Rechners.

Nummer	Programm	Beschreibung	Seite
1000	Speicher- und Schnittstellen-Testprogramm		184
5010	Primfaktoren-Analyse		187
5020	Größter gemeinsamer Teiler, Kleinstes gemeinsames Vielfaches		188
5040	Gleichungssystem (Gauss-Elimination)		189
5050	Quadratische Gleichung		191
5060	Kubische Gleichung		192
5080	Numerische Lösung einer Gleichung (Newton)		194
5090	Numerische Lösung einer Gleichung (Halbierungs-Methode)		196
5100	Matrixoperationen		198
5200	Numerische Integration (Romberg-Methode)		208
5220	Gewöhnliche Differenzialgleichung (Ru.Ku.)		210
5230	Lagrangsche Interpolation		211
5250	Gamma-Funktion		212
5260	Besselsche Funktion $J_n(x)$		213
5270	Besselsche Funktion $Y_n(x)$		214
5280	Modifizierte Besselsche Funktion $I_n(x)$		215
5290	Modifizierte Besselsche Funktion $K_n(x)$		216
5300	Komplexe Zahlen		217
5350	Binär-Dezimal-Hexadezimal		221
5510	Gerade durch zwei Punkte		226
5520	Schnittwinkel von zwei Geraden		227
5530	Abstand zwischen Punkt und Gerade		228
5540	Drehbewegung		229
5550	Kreis durch drei Punkte		230
5560	Länge von Tangenten von einem Punkt zu einem Kreis		231
5570	Tangentialgleichung		232
5600	Dreiecksfläche		234
5605	Fläche eines Trapezoides		235
5610	Fläche eines Parallelogramms		236
5615	Fläche eines Kreises		237
5620	Fläche eines Sektors		238
5625	Fläche eines Segments		239
5630	Fläche einer Ellipse		240
5635	Fläche eines Polygons		240
5650	Flächeninhalt einer Kugel		242

Nummer	Programm	Beschreibung	Seite
5655	Flächeninhalt einer Kugelzone		243
5660	Flächeninhalt eines Kugelsektors		244
5665	Oberfläche eines Kreiszyinders		245
5670	Oberfläche eines Kreiskegels		246
5675	Flächeninhalt eines Kreiskegelstumpfes		247
5700	Rauminhalt einer Kugel		248
5705	Rauminhalt einer Kugelzone		249
5710	Rauminhalt eines Kugelsektors		250
5715	Rauminhalt eines Kreiszyinders		251
5720	Rauminhalt eines Kreiskegels		252
5725	Rauminhalt eines Kreiskegelstumpfes		253
5730	Rauminhalt eines Keils		254
5735	Rauminhalt einer Pyramide		255
5740	Rauminhalt eines Pyramidenstumpfes		256
5745	Rauminhalt eines Ellipsoiden		257
5750	Einbeschriebener Kreis und Umbeschriebener Kreis eines Polygons		258
5760	Regelmäßiger Polyeder		259
5800	Zerlegung in Faktoren		261
5810	Trigonometrische Funktionen		262
5820	Differenziale		263
5830	Integrationen		264
5840	Laplace-Transformation		265
5900	Tabelle des Periodischen Systems		267
5910	Wissenschaftliche Konstanten		272
5920	Konstante der elektrolytischen Dissoziation		274
5930	Bewegung und Energie		275
5932	Wellenbewegung		277
5934	Wechselstrom und Gleichstromkreise		278
5936	Elektrische und magnetische Felder		280
5938	Thermodynamik und Anderes		282
5950	Metrische Umwandlungen für Längen		283
5960	Metrische Umwandlungen für Flächen		285
5970	Metrische Umwandlungen für Rauminhalte		286
5980	Metrische Umwandlungen für Gewicht		288
6210	Obere Wahrscheinlichkeitsintegrale (Normalverteilung)		289
6220	Obere Wahrscheinlichkeitsintegrale ( $x^2$ Verteilung)		290
6230	Obere Wahrscheinlichkeitsintegrale (t Verteilung)		291
6240	Obere Wahrscheinlichkeitsintegrale (F Verteilung)		292
6310	Obere Summenhäufigkeit (Binominal-		293

Nummer	Programm	Beschreibung	Seite
	Verteilung)		
6320	Obere Summenhäufigkeit (Poisson-Verteilung)		294
6330	Obere Summenhäufigkeit (Hypergeometrische Verteilung)		295
6410	Prozentpunkt (Normalverteilung)		296
6420	Prozentpunkt ( $\chi^2$ Verteilung)		297
6430	Prozentpunkt (t Verteilung)		298
6440	Prozentpunkt (F Verteilung)		299
6450	Normale Zufallszahlen		300
6460	Exponentielle Zufallszahlen		301
6500	Statistische Berechnungen mit einer Variablen		302
6510	Lineare Regression ( $y=a+bx$ )		305
6520	Logarithmische Regression ( $y=a+b \ln x$ )		308
6530	Exponentielle Regression ( $y=ab^x$ )		311
6540	Potenz-Regression ( $y=ax^b$ )		314
6610	Mittelwert-Intervallschätzung (für bekannte Varianz)		318
6620	Mittelwert-Intervallschätzung (für unbekannte Varianz)		322
6630	Varianz-Intervallschätzung		326
6640	Standardabweichung-Intervallschätzung		330
6650	Varianzquotient-Intervallschätzung		333
6660	Mittelwertdifferenz-Intervallschätzung		338
6670	Verhältnis-Intervallschätzung		343
6680	Verhältnisdifferenz-Intervallschätzung		344
6710	Grundgesamtheits-Mittelwert-Test (zweiseitig): für bek. Varianz)		346
6711	Grundgesamtheits-Mittelwert-Test (rechtsseitig): für bek. Varianz)		350
6712	Grundgesamtheits-Mittelwert-Test (linksseitig): für bek. Varianz)		354
6720	Grundgesamtheits-Mittelwert-Test (zweiseitig): für unbek. Varianz)		358
6721	Grundgesamtheits-Mittelwert-Test (rechtsseitig): für unbek. Varianz)		361
6722	Grundgesamtheits-Mittelwert-Test (linksseitig): für unbek. Varianz)		364
6730	Grundgesamtheits-Varianz-Test (zweiseitig)		367
6731	Grundgesamtheits-Varianz-Test (rechtsseitig)		370
6732	Grundgesamtheits-Varianz-Test (linksseitig)		373
6740	Varianzquotient-Test (zweiseitig)		376
6741	Varianzquotient-Test (rechtsseitig)		381
6742	Varianzquotient-Test (linksseitig)		386

Nummer	Programm	Beschreibung	Seite
6750	Mittelwertdifferenz-Test (zweiseitig)		390
6751	Mittelwertdifferenz-Test (rechtsseitig)		395
6752	Mittelwertdifferenz-Test (linksseitig)		400
6760	Quotiententest (zweiseitig)		405
6761	Quotiententest (rechtsseitig)		406
6762	Quotiententest (linksseitig)		407
6770	Quotientendifferenztest (zweiseitig)		408
6771	Quotientendifferenztest (rechtsseitig)		410
6772	Quotientendifferenztest (linksseitig)		411

## 12.5 Belegungsplan des Speichers

0000-00FF	Screen memory
0100-01FF	Reserved for internal functions
0200-02FF	INPUT Buffer
0300-03FF	CALCJMP, VALF Buffer
0400-04FF	Reserved for IN/OUT/CALC modes (CALC\$)
0500-074A	Reserved for internal functions
074B-0752	Reserved for storing the user PASSWORD
0753-175A	Reserved for internal functions
175B-175C	Vector pointing to MEMO memory start
175D-1FE4	Reserved for internal functions
1FE5-1FFF	Vectors pointing to P0-P9 memory start
2000-9FFF	User memory (shared by MEMO and P0-p))
A000-BFFF	Repetition of 0000-1FFF
C000-DFFF	Repetition of 0000-1FFF
E000-FFFF	Repetition of 0000-1FFF

Memory area A000-FFFF was either available as user memory (if expansion was present), or would simply repeat the contents of 0000-1FFF

A few glitches are:

POKE 1867,0 ? would delete any user PASSWORD

POKE PEEK(8027)+256\*PEEK(8028),32 ? would recover contents of MEMO after a RESET ALL

POKE PEEK(8027)+256\*PEEK(8028),26 ? would hide contents of MEMO, much like a RESET ALL would do, but without losing the programs

The internal function library was programmed in BASIC itself and could be extracted with a BASIC decompiler.

Any function in the library can be executed from a regular BASIC program by using GOTO "LIB0:NNNN" where NNNN is the function number (e.g. 5010 for prime factorization). The command "GOTO LIB0:0400" executes a self-test program.

Characters 252 to 255 were user defined. They could be defined by issuing the command DEF CHR\$(n)="HHHHHHHHHH" where n ranges from 252 to 255 and the H's are 10 hexadecimal digits (5 bytes). Every byte defines the pixel pattern for a column. Since a column is 7 pixels high, the least significant bit of every byte is ignored.

The CHR\$(26) would activate a different character set for Katakana and Kanji characters. CHR\$(27) would deactivate Kanji.

Via a serial cable the calculator could connect to a PC or to another Casio FX-850P, allowing the transfer to MEMO and stored programs.

## 12.6 Belegungsplan des Erweiterungs-Steckers

### Casio Stecker :

<Draufsicht>



Abbildung 14

Pin	Name	Beschreibung	Bemerkung
1	Vcc	+5 Volt	+5V
2	PB8	RS 232, DSR	Eingang
3	PB7	RS 232, CD	Eingang
4	PB6	RS 232, CTS	Eingang
5	PB5	Nicht benutzt	-
6	PB4	Drucker, BUSY	Eingang
7	PQ1	Nicht benutzt	-
8	IC	Nicht benutzt	-
9	RxD	RS 232, RxD	Eingang
10	PA4	Drucker, INIT	Ausgang
11	PA3	Drucker, STROBE	Ausgang
12	PA2	Drucker, DTR	Ausgang
13	PA1	RS 232, RTS	Ausgang
14	TxD	RS 232, TxD	Ausgang
15	RST0	Reset	Ausgang
16	DB4	Datenbus, Bit 4	Ein/Ausgang
17	DB5	Datenbus, Bit 5	Ein/Ausgang
18	DB3	Datenbus, Bit 3	Ein/Ausgang
19	DB6	Datenbus, Bit 6	Ein/Ausgang
20	DB2	Datenbus, Bit 2	Ein/Ausgang
21	DB7	Datenbus, Bit 7	Ein/Ausgang
22	DB1	Datenbus, Bit 1	Ein/Ausgang
23	DB8	Datenbus, Bit 8	Ein/Ausgang
24	A0	Adressbus, Bit 1	Ausgang
25	A1	Adressbus, Bit 2	Ausgang
26	A2	Adressbus, Bit 3	Ausgang
27	OE	Output Enable	Ausgang
28	WE	Write Enable	Ausgang
29	CS3	CS, Chip Select für FA-6	Ausgang
30	GND	GND (0 Volt bzw. V <sub>CC</sub> T der CPU)	0V

Hinweise:

- Pin 14, 27 und 28 sind im aktiven Zustand LOW.
- Pin 12 steuert beim Speichern auf Kassette den Recorder-Motor (LOW = Motor on)

## 12.7 Befehls-Referenz

### 12.7.1 A

&H Zeichen-Funktion Wandelt den auf &H folgenden 1- bis 4-stelligen Hexadezimalwert in einen Dezimalwert um.

ABS Numerische Funktion Berechnet den Absolutwert des Arguments.

ACS Numerische Funktion Berechnet den Wert des entsprechenden inversen trigonometrischen Funktionswertes für das Argument.

ANGLE Numerische Funktion Spezifiziert die Winkeleinheit.

ASC Zeichen-Funktion Liefert den Zeichencode, der dem ersten Zeichen (ganz links) einer Zeichenfolge entspricht.

ASN Numerische Funktion Berechnet den Wert des entsprechenden inversen trigonometrischen Funktionswertes für das Argument.

ATN Numerische Funktion Berechnet den Wert des entsprechenden inversen trigonometrischen Funktionswertes für das Argument.

### 12.7.2 B

BEEP Grundbefehl Aktiviert den Summer.

### 12.7.3 C

CHR\$ Zeichen-Funktion Liefert ein Zeichen, das dem spezifizierten Zeichencode entspricht.

CLEAR Manueller Befehl Löscht alle Variable und bestimmt die Variablenbereich-Größe entsprechend dem eingegebenen Parameter. Außerdem werden alle offenen Daten geschlossen.

CLOSE Ein/Ausgabe-Befehl Schließt Dateien und deklariert das Ende der Verwendung des I/O (Eingabe/Ausgabe)-Puffers.

CLS Grundbefehl Löscht den Bildschirm.

COS Numerische Funktion Berechnet den Wert des entsprechenden trigonometrischen Funktionswertes für das Argument.

CUR Numerische Funktion Berechnet die Kubikwurzel des Arguments.

### 12.7.4 D

DATA Grundbefehl Nimmt Daten auf, die durch die Anweisung READ gelesen werden sollen.

DEFSEG Grundbefehl Spezifiziert Segment-Grundadressen.

DEG Zeichen-Funktion Wandelt einen Sexagesimalwert in einen Dezimalwert um.

DIM Grundbefehl Deklariert ein Feld.

DMS\$ Zeichen-Funktion Wandelt einen Dezimalwert in eine Sexagesimal-Zeichenfolge um.

### 12.7.5 E

EDIT Manueller Befehl Aktiviert den BASIC-Editor-Modus.

END Grundbefehl Beendet die Programmausführung.

EOF Ein/Ausgabe-Befehl Zeigt das Ende der Datei-Eingabe an.

ERASE Grundbefehl Löscht ein spezifiziertes Feld.

ERL Grundbefehl Liefert die Nummer einer Zeile, in der ein Fehler erzeugt wurde.

ERR Grundbefehl Liefert den Fehlercode, der einem aufgetretenen Fehler entspricht.

EXP Numerische Funktion Berechnet den Wert der Exponentialfunktion für das Argument.

### 12.7.6 F

FACT Numerische Funktion Berechnet die Fakultät des Arguments.

FIX Numerische Funktion Ermittelt den ganzzahligen Anteil des Arguments.

FOR ~ NEXT Grundbefehl Führt die Programmzeilen zwischen der FOR-Anweisung und der NEXT-Anweisung aus und erhöht die Steuervariable, beginnend mit dem Anfangswert. Die Ausführung wird beendet, wenn der Wert der Steuervariablen den spezifizierten Endwert überschreitet.

FRAC Numerische Funktion Ermittelt die Nachkommastellen des Arguments.

FRE Manueller Befehl Liefert die Speicherbereich-Größe entsprechend dem Argument.

### 12.7.7 G

GOSUB Grundbefehl Springt zu einem spezifizierten Unterprogramm.

GOTO Grundbefehl Unbedingte Verzweigung zum spezifizierten Verzweigungsziel.

### 12.7.8 H

HEX\$ Zeichen-Funktion Liefert eine Hexadezimal-Zeichenfolge für einen im Argument spezifizierten Dezimalwert.

HYP ACS Numerische Funktion Berechnet den Wert des entsprechenden inversen Hyperbelfunktion für das Argument.

HYP ASN Numerische Funktion Berechnet den Wert des entsprechenden inversen Hyperbelfunktion für das Argument.

HYP ATN Numerische Funktion Berechnet den Wert des entsprechenden inversen Hyperbelfunktion für das Argument.

HYP COS Numerische Funktion Berechnet den Wert des entsprechenden Hyperbelfunktion für das Argument.

HYP SIN Numerische Funktion Berechnet den Wert des entsprechenden Hyperbelfunktion für das Argument.

HYP TAN Numerische Funktion Berechnet den Wert des entsprechenden Hyperbelfunktion für das Argument.

### 12.7.9 I

IF ~ THEN ~ ELSE

IF ~ GOTO ~ ELSE Grundbefehl Führt die THEN-Anweisung oder GOTO-Anweisung aus, wenn die spezifizierte Bedingung erfüllt wird. Die ELSE-Anweisung wird ausgeführt, wenn die spezifizierte Bedingung nicht erfüllt wird.

INKEY\$ Grundbefehl Weist eine einzelne Zeicheneingabe von der Tastatur einer Variablen zu.

INPUT Grundbefehl Weist Daten, die über die Tastatur eingegeben werden, Variablen zu.

INPUT# Ein/Ausgabe-Befehl Liest Daten aus einer sequentiellen Datei.

INPUT\$ Grundbefehl Weist eine spezifizierte Anzahl Zeichen von der Tastatur einer Variablen zu.

INPUT\$ Ein/Ausgabe-Befehl Liest die spezifizierte Anzahl Zeichen aus einer sequentiellen Datei.

INT Numerische Funktion Ermittelt die größte ganze Zahl, die den Wert des Arguments nicht überschreitet.

**12.7.10 L**

LEFT\$ Zeichen-Funktion Liefert eine spezifizierte Anzahl Zeichen vom linken Ende einer Zeichenfolge.

LEN Zeichen-Funktion Liefert einen Wert, der die Anzahl der Zeichen einer Zeichenfolge angibt.

LET Grundbefehl Weist den Wert eines Ausdrucks an der rechten Seite einer Gleichung der Variablen an der linken Seite zu.

LIST

LIST ALL Manueller Befehl Zeigt das gesamte oder einen Teil des gegenwärtigen spezifizierten Programms an.

LIST# Datenbank-Befehl Zeigt alle Datenbank-Daten an.

LLIST Ein/Ausgabe-Befehl Ausgabe von Programminhalten an den Drucker.

LLIST# Datenbank-Befehl Gibt alle Datenbank-Daten auf dem Drucker aus.

LN Numerische Funktion Berechnet den Wert der entsprechenden Logarithmusfunktion für das Argument

LOAD

LOAD ALL Ein/Ausgabe-Befehl Liest aus einer Datei in den Speicher.

LOAD# Datenbank-Befehl Liest Daten in den Datenbank-Bereich.

LOCATE Grundbefehl Bewegt den Cursor zu einer spezifizierten Position auf dem virtuellen Bildschirm.

LOG Numerische Funktion Berechnet den Wert der entsprechenden Logarithmusfunktion für das Argument

LPRINT Ein/Ausgabe-Befehl Ausgabe von text an den Drucker.

**12.7.11 M**

MID\$ Zeichen-Funktion Liefert eine spezifizierte Anzahl Zeichen von einer spezifizierten Position in einer Zeichenfolge.

**12.7.12 N**

NCR Numerische Funktion Berechnet die Kombination  $nCr$  für die Werte von  $n$  und  $r$ .

NEW

NEW ALL Manueller Befehl Löscht ein Programm.

NEW# Datenbank-Befehl Löscht Datenbank-Daten.

NPR Numerische Funktion Berechnet die Permutation  $nPr$  für die Werte von  $n$  und  $r$ .

**12.7.13 O**

ON ERROR GOTO Grundbefehl Spezifiziert die Zeilennummer, zu der die Ausführung verzweigt wird, wenn ein Fehler auftritt.

ON GOSUB Grundbefehl Sprung zum spezifizierten Unterprogramm entsprechend der spezifizierten Verzweigungsbedingung.

ON GOTO Grundbefehl Springt zum spezifizierten Verzweigungsziel entsprechend der spezifizierten Verzweigungsbedingung.

OPEN Ein/Ausgabe-Befehl Eröffnet eine Datei zur Verarbeitung.

**12.7.14 P**

PASS Manueller Befehl Spezifiziert oder löscht ein Passwort.

PEEK Grundbefehl Liefert den Wert, der in der spezifizierten Speicheradresse gespeichert ist.

PI Numerische Funktion Gibt den Wert von  $\pi$  an.

POKE Grundbefehl Schreibt Daten in eine spezifizierte Adresse.

POL Numerische Funktion Wandelt kartesische Koordinaten  $(x, Y)$  in Polkoordinaten  $(r, \theta)$  um.  
 PRINT Grundbefehl Zeigt Daten auf dem Bildschirm an.  
 PRINT# Ein/Ausgabe-Befehl Ausgabe von Daten in eine sequentielle Datei.

### 12.7.15 R

RAN# Numerische Funktion Erzeugt einen Zufallswert im Bereich von 0 bis 1.  
 READ Grundbefehl Liest den Inhalt der DATA-Anweisung in den Speicher.  
 READ# Datenbank-Befehl Liest Daten aus dem Datenbank-Bereich.  
 REC Numerische Funktion Wandelt Polkoordinaten  $(r, \theta)$  in kartesische Koordinaten  $(x, Y)$  um.  
 REM ( ' ) Grundbefehl Ermöglicht das Einfügen von Anmerkungen oder Kommentaren in ein Programm. Dieser Befehl wird nicht ausgeführt.  
 RESTORE Grundbefehl Spezifiziert eine DATA-Zeile zum Lesen durch die READ-Anweisung.  
 RESTORE# Datenbank-Befehl Sucht bestimmte Daten im Datenbank-Bereich und verändert die Lesereihenfolge von Datenbank-Daten.  
 RESUME Grundbefehl Bewirkt Rückkehr aus einer Fehlerabwicklungsroutine zum Hauptprogramm.  
 RETURN Grundbefehl Rückkehr der Ausführung von einem Unterprogramm zum Hauptprogramm.  
 RIGHT\$ Zeichen-Funktion Liefert eine spezifizierte Anzahl Zeichen von der rechten Seite einer Zeichenfolge.  
 ROUND Numerische Funktion Rundet das Argument an der spezifizierten Stelle.  
 RUN Manueller Befehl Führt ein Programm aus.

### 12.7.16 S

SAVE  
 SAVE ALL Ein/Ausgabe-Befehl Sichert ein Programm in eine spezifizierten Datei.  
 SAVE# Datenbank-Befehl Gibt Datenbank-Daten zu einer Datei aus, die durch den Dateideskriptor spezifiziert wurde.  
 SET Grundbefehl Spezifiziert das Ausgabeformat von numerischen Daten.  
 SGN Numerische Funktion Ermittelt den Wert, der dem Vorzeichen des Arguments entspricht.  
 SIN Numerische Funktion Berechnet den Wert des entsprechenden trigonometrischen Funktionswertes für das Argument.  
 SQR Numerische Funktion Berechnet die Quadratwurzel des Arguments.  
 STOP Grundbefehl Unterbricht die Programmausführung.  
 STR\$ Zeichen-Funktion Wandelt das Argument (numerischer Wert oder Wert eines numerischen Ausdrucks) in eine Zeichenfolge um.

### 12.7.17 T

TAB Grundbefehl Gibt eine Horizontal-Tabulatorspezifikation zum Bildschirm oder Drucker aus.  
 TAN Numerische Funktion Berechnet den Wert des entsprechenden trigonometrischen Funktionswertes für das Argument.  
 TROFF Manueller Befehl Hebt den Ablaufverfolgungs-Modus auf.  
 TRON Manueller Befehl Spezifiziert den Ablaufverfolgungs-Modus.

**12.7.18 V**

VAL Zeichen-Funktion Wandelt eine numerische Zeichenfolge in einen numerischen Wert um.

VALF Zeichen-Funktion Führt mit einem numerischen Ausdruck, der als Zeichenfolge ausgedrückt ist, eine Berechnung aus und liefert das Ergebnis.

VARLIST Manueller Befehl Zeigt Variablennamen und Feldnamen an.

VERIFY Ein/Ausgabe-Befehl Überprüft den Inhalt einer auf Kassette gespeicherten Datei.

**12.7.19 W**

WRITE# Datenbank-Befehl Überschreibt und löscht Datenbank-Daten.

## **12.8 Index**

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
_
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
_
```

```
CAPS S CAL BASIC DEGRADGRA MEMO IN EDIT LIB 8 8 8 8 DEFM PRT TR STOP
12345678901234567890123456789012
12345678901234567890123456789012
```

**Listing 7** XXXX

**Prg. 12.1** XXXX

**LIB**

Abbildung 15

Zeile	Beschreibung

Piktogramme:



**Beispiel**



**Hinweis**



**Achtung**



Überblicksinformationen



Handlungsanweisungen



Hintergrundinformationen



Hinweis bzw. Frage



# Casio Basic Tutorial 1.00

## Was Sie schon immer über Casio Basic wissen wollten:

- Einführung
- Schnelleinstieg
- Voraussetzungen
- Grundlagen
- Mit Basic rechnen
- Variablen speichern Zahlen oder Texte
- Tipps für das Programmieren
- Entscheidungen treffen
- Anweisungen wiederholen



### **Autor**

Dipl.-Ing.(FH)  
Manfred Becker

E-Mail: [mani.becker@web.de](mailto:mani.becker@web.de)

URL: <http://manib.ma.funpic.de>

## Hallo liebe Casio Fans,

Dieses Tutorial lehrt Sie die Grundlagen von Casio BASIC, der Programmiersprache des Taschenrechners **CASIO FX-850P**.

Sie benötigen keinerlei Vorkenntnisse, um mit diesem Tutorial das Programmieren zu lernen.

Viel Spass dabei!

<http://manib.ma.funpic.de>

ISBN-10: 3-123-45678-0  
ISBN-13: 978-3-123-45678-0

