Einführung in den TI-89 Titanium

Ivo Blöchliger

Version vom 29. Juni 2009
anlässlich der Spezialwoche Akzentfach Mathematik
an der Kantonsschule Wohlen AG

Inhaltsverzeichnis

1 Variablen

Variablen auf dem TI-89 bestehen aus 1 bis 8 Buchstaben (wobei ab dem zweiten Buchstaben auch Zahlen verwendet werden können). Gross- und Kleinschreibung werden ignoriert.

Variablen sind entweder undefiniert und können für algebraische Umformungen gebraucht werden, oder sie sind definiert und werden durch ihren Wert ersetzt. Ausnahmen sind Ausdrücke, wo es klar ist, dass man den Platzhalter und nicht den Wert meint, wie z. B. bei Gleichungen oder Funktionsdefinitionen.

1.1 Speichern

Einer Variable kann mit der STO-Taste ein Wert zugewiesen werden: `15→x` speichert 15 in der Variablen x ab. In allen Ausdrücken wird von jetzt an x durch 15 ersetzt. Z. B. gibt `x^2` das Resultat 225 aus.

1.2 Löschen

Um formale Berechnungen durchzuführen, ist es nicht wünschenswert, dass Variablen durch Werte ersetzt werden. Dazu müssen die gewünschten Variablen gelöscht werden.

Mit F6 "clear a-z" werden alle Variable gelöscht, die nur aus einem Buchstaben bestehen.

Eine einzelne Variable kann mit dem Befehl `DelVar` gelöscht werden. Z. B. löscht `DelVar asdf` die Variable `asdf`.

1.3 VAR-LINK

Mit 2ND und − wird das "VAR-LINK"-Menu aufgerufen, wo die Variablen verwaltet werden können. Dort können Variablen gelöscht, umbenannt, in andere Ordner verschoben werden oder auf andere Rechner übertragen werden.

1.4 Ordner

Alle Variablen (und Programme und Funktionen) sind in einem bestimmten Ordner abgelegt. Der Standardordner heisst `main`. Der aktive Ordner wird ganz unten links in der Statuszeile angezeigt. Ordner können im VAR-LINK angelegt und verwaltet werden.

Den aktiven Ordner kann man über MODE und "Current Folder" festlegen.

Will man eine Variable (bzw. Programm oder Funktion) aus einem anderen Ordner als dem aktiven Ordner verwenden, muss der Ordnername der Variable vorangestellt werden. Z. B. `amat\delta` um die Variable `delta` im Ordner `amat` aufzurufen. Die effiziente Eingabe kann über den VAR-LINK erfolgen.

1.5 Systemvariablen

Gewisse Variablen werden auch vom System gebraucht und können in dem meisten Fällen auch verändert werden. Z. B. sind die Variablen `xmin`, `xmax`, `ymin` usw. für die Grafikausgabe zuständig. Mehr dazu im Window-Editor ◆ F2.

1.6 Funktionen speichern

Funktionen können z. B. wie folgt definiert werden: `x^2→f(x)`. Dann ergibt z. B. `f(5)` das Resultat 25.

Es können auch Funktionen mit mehr als einer Variable definiert werden, wie z. B. eine Funktion, die das Kapital mit Zinseszins auf einem Bankkonto nach n Jahren berechnet: `k*(1+p)^n →f(k,p,n)`. Damit liefert z. B. `f(3400, 0.03, 5)` das Resultat 3941.53, das Kapital von 3400.– zu 3% während 5 Jahren verzinst.

1.7 Aufgaben

Aufgabe 1 Legen Sie einen Ordner mit Namen `amat` an. Aktivieren Sie diesen Ordner (so dass im Home unten links `amat` erscheint.

Aufgabe 2 Definieren Sie eine Funktion `p(x,y)`, die den Abstand vom Punkt $P = (x, y)$ zum Ursprung $O = (0, 0)$ berechnet. Kontrolle: `p(-3,4)` liefert 5.

Aufgabe 3 Löschen die Variablen `a` bis `z`. Definieren Sie eine allgemeine quadratische Funktion $f(x) = ax^2 + bx + c$. Werten Sie die Funktion an den Stellen -1, 0 und 1 aus. Speichern Sie nun die Werte $\frac{1}{2}$, -1 und 2 in die Variablen `a`, `b` und `c` und werten Sie die Funktion nochmals aus.

2 Der Katalog

Müssen Befehle eingegeben werden, ist man oft schneller, den CATALOG zu benutzen. Drücken sie im

Katalog jeweils den Anfangsbuchstaben des gesuchten Befehls (wobei die Taste [ALPHA] nicht nötig ist). Mit [2ND] [⌃] bzw. mit [2ND] [⌄] können sie schneller durch die Liste gehen.
Mit [CATALOG] [F4] erhalten Sie eine Liste mit benutzerdefinierten Funktionen und Programmen.

2.1 Das MATH-Menu

Mit [2ND] [5] öffnen Sie das Math-Menu, über das man bestimmte Befehle noch schneller erreicht (durch Drücken der entsprechenden Zahlen in den Menus und Untermenus).

2.2 Das CUSTOM-Menu

Es kann auch selbst ein Menu mit den wichtigsten Befehlen definiert werden. Damit können alle wichtigen Befehle für ein bestimmtes Thema gruppiert und abrufbereit gehalten werden. Wie man ein solches Menu anlegt, siehe Abschnitt 6.4.

3 Datentypen

Der TI-89 kennt verschiedene Datentypen, wie z. B. Dezimalbrüche, algebraische Ausdrücke und Vektoren. Weiter kennt der Rechner auch Zeichenketten, "Strings" genannt, Listen und Matrizen.

3.1 Exakte Zahlen und Dezimalbrüche

Ist der Taschenrechner im AUTO-Modus, versucht der Rechner, wenn möchlich ein exaktes Resultat zu erhalten (z. B. $\pi\sqrt{2}$). Ist dies nicht möglich, werden Dezimalbrücke als Näherung ausgegeben. I Die Näherung kann auch erzwungen werden, indem [◆] [ENTER] gedrückt wird (≈). Eine Näherung kann auch erzwungen werden, indem man in der Eingabe irgendwo einen Dezimalbruch eingibt, oder z. B. 0.0 addiert.

3.2 Zeichenketten (Strings)

Zeichenketten werden zwischen Anführungszeichen geschrieben (z. B. `"hallo"→s` speichert das Wort "hallo" in der Variablen `s`.

Mittels `mid(s,2,1)` kann der Buchstabe an Position 2 extrahiert werden, also `"a"`. Die 1 bezeichnet die Länge.

Mittels `ord(s)` bekommt man die Nummer (den sogenannten ASCII-Code) des ersten Buchstaben im String `s`.

Mittels `char(113)` erhält man den String bestehend aus dem Buchstaben mit dem ASCII-Code 113, in diesem Fall `"q"`.

Mittels `dim(s)` erhält man die Länge vom String `s`.

Mittels `string(Zahl)` kann eine Zahl in eine Zeichenkette umgewandelt werden. Z.B. liefert `string(113)` die Zeichenkette `"113"`.

Mittels `&` können zwei Zeichenketten aneinandergehängt werden. Z.B. liefert `"Hallo" & "Welt"` die Zeichenkette `"HalloWelt"`.

Aufgabe 4 Ermitteln Sie die ASCII-Codes von Gross- und Kleinbuchstaben. Was stellen Sie fest? Wie gross ist die Differenz? Warum könnte das praktisch sein?

Aufgabe 5 Wie viele Buchstaben im Alphabet gibt es von und mit 'H' bis und mit 'S'? Lösen Sie die Aufgabe mit dem Taschenrechner.

Aufgabe 6 Welchem Buchstaben (bzw. Zeichen) entspricht der ASCII-Code 55?

3.3 Listen

Listen können Zahlen, algebraische Ausdrücke und Strings enthalten (leider keine Matrizen oder Listen).

Listen werden mit geschweiften Klammern und Kommata geschrieben. Z. B. `{2,3,5,7,11,13}→t`.

Mit Listen kann gerechnet werden, wie mit Zahlen. Der Vorteil ist, dass alle Berechnungen auf einmal ausgeführt werden. So liefert z. B. `t^2` die Liste `{4,9,25,49,121,169}`.

Werden zwei gleichgrosse Listen miteinander verrechnet (z. B. multipliziert) geschehen die Operationen elementweise und das Resultat ist ebenfalls wieder eine gleich grosse Liste.

Es kann auch mit eckigen Klammern auf einzelne Elemente zugegriffen werden. Z. B. liefert `t[3]` die Zahl 5. Das erste Element hat die Nummer 1. Die Länge einer Liste kann mit `dim(t)` erfragt werden.

Einzelne Elemente können ebenfalls verändert werden. Z. B. ändert `97→t[3]` das dritte Element auf 97.

3.3.1 Listen erzeugen mit seq

Der Befehl `seq(2*i,i,0,5)` erzeugt die Liste {0,2,4,6,8,10}.
Das erste Argument ist ein mathematischer Ausdruck, das zweite die Laufvariable (die jeweils in Einerschritten erhöht wird), das dritte und vierte Argument geben die untere und obere Grenze der Laufvariablen an.

3.3.2 Wichtige Operationen auf Listen

Im Folgenden nehmen wir an, dass in `l` und `m` eine Liste gespeichert ist.

`dim(l)` ergibt die Anzahl Elemente.

`sum(l)` ergibt die Summe aller Elemente.

`sum(l,a,b)` ergibt die Summe der Elemente Nummer a bis Element Nummer b.

`product(l,a,b)` ergibt das Produkt der Elemente Nummer a bis Element Nummer b.

`augment(l,m)` ergibt eine Liste mit den Elementen von `l`, gefolgt von den Elementen von `m`.

`min(l)` **und** `max(l)` ergibt das kleinste bzw. grösste Element.

`SortA l` **und** `SortD l` Sortiert die Liste `l` aufsteigend (**A**scending) oder absteigend (**D**escending). Achtung: Die Liste `l` wird verändert!

`SortA l,m` **und** `SortD l,m` Sortiert die Liste `l` und nimmt bei der Liste `m` die gleichen Umplatzierungen vor.

3.4 Aufgaben

Aufgabe 7 Erzeugen Sie eine Liste `x` mit allen ganzen Zahlen zwischen -3 und $+3$. Wie sehen die Funktionswerte von $g(x) = \frac{1}{2}x^2 - 2$ in diesen Punkten aus? Erzeugen Sie eine Liste `y`, die die Funktionswerte enthält. Sortieren Sie am Schluss die Liste `x` aufsteigend und parallel dazu die Liste `y`. Indem Sie beide Listen anzeigen, lesen die möglichen Koordinaten für den Scheitelpunkt des Graphen von $g(x)$ ab.

Aufgabe 8 Erzeugen Sie eine Liste aller Zweierpotenzen bis 2^{10}. Bilden Sie die Summe über diese Elemente auf verschiedene Arten.

Aufgabe 9 Erzeugen Sie eine Liste `a` mit allen Vielfachen von 15 zwischen 0 und 360.
Überprüfen Sie, dass $(\cos(\alpha))^2 + (\sin(\alpha))^2 = 1$ für alle Winkel der Liste `a`.

Aufgabe 10 Mit `seq(rand(1000),i,1,21)→w` erzeugen Sie 21 Zufallszahlen zwischen 1 und 1000. Bestimmen Sie das kleinste und grösste Element der Liste `w`. Bestimmen Sie das 5.-grösste und 11.-grösste Element.

Aufgabe 11 Senden Sie Ihre Liste mit Zufallszahlen aus Aufgabe 10 an den Rechner Ihrer Banknachbarin oder Ihres Banknachbars.

Aufgabe 12 Erzeugen Sie eine Liste mit allen Kleinbuchstaben von 'a' bis 'z'.

Aufgabe 13 Erzeugen Sie eine Liste mit 5 zufälligen Grossbuchstaben.

4 Solve, der Alleskönner

Mit `solve` lassen sich sowohl Gleichungen mit einer Variablen lösen, z. B. `solve(x^2+x=6,x)`, als auch Gleichungssysteme, z. B. `solve(x+y=2 and x-2*y=1,{x,y})`. Die Variablen eines Gleichungssystems werden als Liste aufgeführt.
Die aktuelle Version vom Taschenrechner[1] versteht auch vektorielle Gleichungen. Die unbekannten Grössen sind aber immer Skalare (Zahlen, bzw. mathematische Ausdrücke, die für Zahlen stehen). Eine unbekannte kann also nicht für einen Vektor stehen. Ein unbekannter Vektor wäre dann z. B. [x,y,z] und sucht dann diese drei Unbekannten.

4.1 Unendlich viele Lösungen

Trigonometrische Gleichungen haben i. A. unendlich viele Lösungen. Der Rechner gibt diese aus. Beispiel: `solve(cos(x)=1/2,x)` liefert `x=60·(6·@n15+1) or x=60·(6·@n15-1)` also die Winkel $\pm 60°$ plus alle beliebige, ganzzahlige Vielfache

[1] OS Version 3.10, 7/18/2005. Sie finden Ihre Version mit HOME F1 ALPHA =

von 360°. (Die Notation `@n15` steht für eine beliebige ganze Zahl).

4.2 Extrahieren der Lösungen mit `exp▷list`

Das Format der Lösungen einer Gleichung ist wieder eine Gleichung. Das ist oft unpraktisch, wenn man des Resultat weiter verwenden möchte. Kopieren wäre ein Ausweg, was aber in einem Programm nicht möglich ist.

Mit `exp▷list(solve(x^2+x=6,x),x)` erhält man eine Liste der Lösungen, auf die mit eckigen Klammern zugegriffen werden kann.

Eine Alternative bietet die Funktion `zeros`. Dazu muss die Gleichung allerdings umgeformt werden, indem man alles auf eine Seite nimmt. Diese Seite kann dann als Funktion aufgefasst werden und man sucht nun die Nullstellen dieser Funktion: `zeros(x^2+x-6,x)`. Der TR liefert direkt eine Liste mit den Werten.

4.3 Einschränken der Lösungsmenge

Sucht man z. B. nur positive Lösungen einer Gleichung kann man weitere Bedingungen formulieren: `solve(x^2+x=6 and x>0,x)`.

4.4 Ungleichungen

`solve(1/(2*x-1)+x/(x-2)≥1,x)` liefert eine Lösung mit der Warnung, das die Lösungmenge eventuell noch grösser sein könnte. In diesem Fall ist Ihre Expertise gefragt, um das Resultat zu überprüfen.

Hinweis: Das $\geq$ erhalten Sie am schnellsten über [2ND] [5] [8] [3].

4.5 Aufgaben

Aufgabe 14 Finden Sie alle Lösungen der Gleichung $\sin(x) = \cos(x)$. Interpretieren Sie das Resultat und sehen Sie ein, dass Sie das ohne TR viel schneller hätten lösen können.

Aufgabe 15 Erzeugen Sie eine Liste mit den Lösungen der Gleichung: $x^3 + 11x = 6(x^2 - 1)$

5 Funktionsgraphen

Überprüfen Sie, durch drücken von [MODE], dass "Graph" auf "FUNCTION" gesetzt ist.

5.1 Eingabe

Über [◆] [F1] gelangen Sie in den "Y-Editor", wo Sie Funktionsgleichungen festlegen können und durch setzen der Häkchen ([F4]) bestimmen können, welche gezeichnet werden sollen.

5.2 Zeichnen

Mit [◆] [F3] kommt man in den Grafik-Modus. Wahrscheinlich ist die Einstellung des Koordinatensystems noch nicht so, wie Sie gerne hätten.

5.3 Window-Einstellungen

Mit [◆] [F2] gelangen Sie in den Window-Editor, wo sie den Ausschnitt des Koordinatensystems, der auf Bildschirm dargestellt werden soll, genau beeinflussen können (mit `xmin`, `xmax` für den horizontalen und `ymin`, `ymax` für den vertikalen Bereich).

Mit `xscl` und `yscl` beeinflussen Sie Abstände der Striche auf den Achsen.

5.4 Zoom-Möglichkeiten

Die wichtigste Zoom-Möglichkeit ist `ZoomSqr`. Diese Funktion passt einen Bereich so an, dass die Skalierung der beiden Achsen identisch ist (d.h. Kreise erscheinen auch als solche, und nicht als Ellipsen). Welcher der beiden Bereiche angepasst wird, ist allerdings nicht klar :-(

Zur Funktionsweise der weiteren Zoom-Funktionen lesen Sie bitte das Handbuch.

5.5 Weitere Grafik-Einstellungen

Drücken Sie [◆] [|] im Graph-Modus.

5.6 Zeichnen Unterbrechen

Drücken Sie einfach [ON].

5.7 Aufgaben

Aufgabe 16 Zeichnen Sie die Funktionen $f(x) = x - x^2$ und $g(x) = x^2 - 3x + 2$. Wählen Sie den Zoom so, dass die beiden "schön" dargestellt werden.

Aufgabe 17 Die in Aufgabe 16 definierten Funktionen können Sie mit `y1` und `y2` verwenden. Z. B. liefert `y1(t)` den Ausdruck `t-t^2`. Beweisen Sie mit dem TR (und etwas Überlegen), dass sich die beiden Funktionsgraphen aus Aufgabe 16 genau berühren.

Aufgabe 18 Zeichnen Sie die Graphen von $a(x) = |x| - \sqrt{1-x^2}$ und $b(x) = |x| + \sqrt{1-x^2}$. Passen Sie den Windowbereich entsprechend an. Schalten Sie die Achsen aus und hinterlegen Sie das Bild mit Gitterpunkten.

Aufgabe 19 In Gruppenarbeit, wählen Sie eine "Standardaufgabe" aus einem Unterrichtsfach, die sich möglichst effizient mit dem TR lösen lassen soll. Präsentieren Sie dann kurz diese Aufgabe vor Publikum.

6 Programmierung

Um schneller in den Programm-Editor zu kommen, wird empfohlen den Application Desktop abzuschalten: [MODE] [F3] und setzen Sie dort "Apps Desktop" auf "OFF".

6.1 Erstellen eines Programmes

- Drücken Sie [APPS] und öffnen Sie dann den "Programm Editor" (mit [7]).
- Wählen Sie "New".
- Unter "Variable", geben Sie den Namen des Programmes ein. Für das erste Beispiel geben Sie `ekreis` als Name ein.

Jetzt können Sie beginnen Ihr erstes Programm zu schreiben.

Tipp: Drücken Sie [2ND] [APPS] um zwischen Home und Programm Editor hin und her zu wechseln.

6.1.1 Navigation im Programmtext

[2ND] [Pfeiltaste] An den Anfang/Schluss der Zeile, bzw. einen Bildschirm nach oder unten springen.

[↑] [Pfeiltaste] Text markieren.

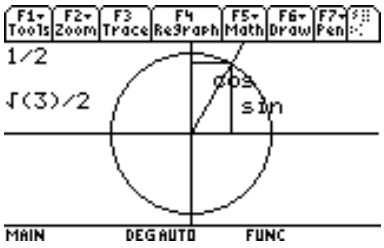
[◆] [↑] Markierten Text kopieren.

[◆] [ESC] Kopierten Text einfügen.

[◆] [⌃] **und** [◆] [⌄] An den Anfgang/Ende des Programmtextes springen.

6.2 Trigo im Einheitskreis

Ziel ist es, ein Programm `ekreis(w)` zu schreiben, das Cosinus und Sinus im Einheitskreis für den Winkel `w` einzeichnet. Z. B. soll `ekreis(60)` folgendes Bild liefern:



- Legen Sie ein neue Programm `ekreis` an (bzw. öffnen Sie es).
- Die erste Zeile soll auf `ekreis(w)` ergänzt werden (`w` ist der Winkel, zu dem die Skizze gezeichnet werden soll).
- Zwischen `Prgm` und `EndPrgm` fügen Sie, je auf einer Zeile, folgende drei Befehle ein (benutzen Sie den [CATALOG]!): `FnOff`, `ClrDraw`, `Circle 0,0,1`.

Drücken Sie [HOME]. Um das Programm zu starten, schreiben Sie entweder von Hand `ekreis(60)`, oder Sie drücken [CATALOG] [F4] [e] und wählen Ihr Programm aus.

Sie sollten nun einen (mehr oder weniger grossen) Kreis auf dem Bildschirm haben. Passen Sie die Windowparameter an, so dass der Einheitskreis schön auf den Bildschirm passt und starten das Programm nochmals.

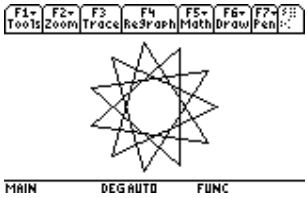
Lesen Sie im Handbuch nach, was die Befehle `FnOff`, `ClrDraw` und `Circle 0,0,1` genau bewirken.

Vervollständigen Sie das Programm. Nützliche Befehle: `Line`, `PtText`, `string` und `&`.

Schreiben Sie gute Window-Einstellungen direkt ins Programm (mit z. B. $-1.1 \rightarrow$`ymin`).

6.3 Sternefoifi!

Ziel ist es, Sterne wie folgt zu zeichnen:



Legen Sie ein neue Programm mit Name `stern` an. Die Anzahl und Art der Zacken soll mit Parametern festgelegt werden können.

6.3.1 For-Schlaufe

Um Dinge zu wiederholen, bietet sich die For-Schlaufe an:

```
For i,1,10
```
hier der Programmcode, der wiederholt werden soll,

wobei die Variable i alle Werte von 1 bis 10 annimmt.

```
EndFor
```

For-Schleifen werden am besten mit F2 4 eingegeben!

6.3.2 Variablen lokal definieren

Definieren Sie alle Variablen (hier z. B. `i` und `j`) am Anfang des Programms mit `Local i,j`. Damit beeinflussen sich die Variable `i` im Programm und die Variable `i` im Home nicht, was unerwünschte Nebeneffekte zur Folge haben könnte. Parameter, die übergeben werden, sind automatisch lokal.

6.3.3 Code einrücken

Rücken Sie den Code innerhalb der For-Schlaufe konsequent um 2 Leerschläge (ALPHA (-))ein. Das erhöht die Lesbarkeit des Codes ungemein. Dasselbe gilt natürlich für andere Schlaufen oder bedingte Programmblöcke (zwischen `If` und `EndIf`).

6.4 CUSTOM-Menu definieren

Legen Sie ein neues Programm an, z. B. `mymenu`. Das Menu wird wie folgt definiert:

```
:mymenu()
:Prgm
:Custom
:Title "Vektor"
:Item "norm("
:Item "dotP("
:Title "Amat"
:Item "rand("
:Item "mymenu()"
:EndCustm
:CustmOn
:EndPrgm
```

Es können natürlich noch mehr "Item" und "Title" stehen. Um das Menu zu aktivieren muss zuerst das Programm `mymenu()` im HOME gestartet werden. Danch kann das Menu mit 2ND HOME ein- und ausgeschaltet werden. Wird das Menu verändert, muss zuerst wieder das Programm gestartet werden.

Es können natürlich mehrere CUSTOM-Menus definiert werden. Es muss dann einfach das entsprechende Programm aufgerufen werden (die sich idealerweise dann ebensfalls in den entsprechenden Menus befinden).

6.5 Caesar Verschlüsselung

Die Caesar Verschlüsselung besteht darin, dass in einem Text alle Buchstaben im Alphabet um eine bestimmte Anzahl Positionen verschoben werden. Z. B. auch "bla" wird "cmb" (Verschiebung um 1 Position).

6.5.1 Funktionen

Es soll eine *Funktion* (im Gegensatz zu einem Programm) geschrieben werden, die einen String (Zeichenkette) verschlüsseln kann. Eine Funktion (wie in der Mathematik) liefert einen Wert zurück. Dies wird mit dem Befehl `Return` erledigt.

Im Gegensatz zu Programmen *müssen* alle verwendeten Variablen als *lokal* definiert werden. Auch dürfen in einer Funktion keine globalen Variablen verändert werden und gewisse Befehle dürfen nicht aufgerufen werden.

6.5.2 Hinweise zur Programmierung

Legen Sie eine neue `Function` an. Nennen Sie sie `caesar`. Die Funktion soll zwei Parameter übernehmen: die zu verschlüsselnde Zeichenkette und die Anzahl Positionen, um die das Alphabet verschoben werden soll.

Die Programmlogik könnte wie folgt aussehen:

- Zuerst wird eine lokale Variable mit der leeren Zeichenkette (Nullstring) initialisiert. Z. B. `""→r`.
- In einer `For`-Schlaufe wird jeder Buchstabe vom zu verschlüsselnden Wort durchgegangen, der Buchstabe verschoben und der Variable `r` wie folgt angehängt: `r&`*Buchstabe*`→r`.
- Am Schluss wird das Resultat mit `Return r` zurückgegeben.

Versuchen Sie dann Zeichenketten zu verschlüsseln, und diese auch wieder zu entschlüsseln. Verbessern Sie das Programm!

6.6 Die `If`-Abfrage

`If`-Abfragen werden verwendet, wenn ein bestimmter Programmteil nur dann ausgeführt werden soll, wenn eine bestimmte Bedingung wahr ist.

Die einfachste `If`-Abfrage sieht wie folgt aus:

`If` *B*`edingung Then`
Beliebig viele Zeilen Programmcode
`EndIf`

Es können noch beliebig viele `ElseIf`-Abfragen folgen, und eine `Else`-Verzweigung, die genau dann ausgeführt wird, wenn keine der oberen Bedingungen wahr ist.

`If` *Bedingung* `Then`
Beliebig viele Zeilen Programmcode
`ElseIf` *Bedingung* `Then`
Beliebig viele Zeilen Programmcode
`Else`
Beliebig viele Zeilen Programmcode
`EndIf`

Für die Caesar-Verschlüsselung sollten sie "Buchstaben" vor dem 'a' und nach dem 'z' abfangen und entsprechend korrigieren.

Aufgabe 20 Entschlüsseln Sie folgdene Zeichenkette: **oldrpseklnvklnv**.

Aufgabe 21 Suchen Sie, möglichst effizient, kurze Wörter, die geeignet Caesar-verschlüsselt andere Wörter ergeben.

6.7 Standardaufgaben dem Rechner beibringen

Als Beispiel soll das Lösen der quadratischen Gleichung programmiert werden. Das Programm soll die (am besten lokal definierten) Variablen `a`, `b` und `c` einlesen (die Koeffizienten der Gleichung), die Diskriminante bestimmen, die Formel dafür ausgeben, die Anzahl Lösungen und die Lösungen selbst.

Dazu stehen Ihnen (unter anderem) die Befehle `Input`, `Disp`, `Output` und `Pause` zur Verfügung.

6.8 Zufallszahlen

Die Funktion `rand()` liefert eine "reelle" Zufallszahl zwischen 0 und 1. `rand(20)` liefert eine natürliche Zufallszahl zwischen 1 und 20 (inklusive).

Mittels z. B. `RandSeed 123` kann der Zufallsgenerator initialisiert werden. (Die Zahl 123 kann auch anders gewählt werden.) Nach einem solchen Aufruf, wird immer die gleiche Sequenz von Zahlen erzeugt. Sollen die Zahlen "zufälliger" erzeugt werden, initialisieren sie am Besten mit `RandSeed getTime()[3]` oder noch besser mit `RandSeed sum({3600,60,1}*getTime())`. Frage: Was stellt die berechnete Zahl effektiv dar?

Aufgabe 22 Schreiben Sie ein Programm, das eine Zufallszahl zwischen 1 und 100 erzeugt. Der Benutzer soll dann die Zahl erraten, wobei das Programm jeweils sagt, ob die gesuchte Zahl grösser oder kleiner ist. Das Programm soll auch ausgeben, wie viele Versuche der Benutzer benötigt hat.

Aufgabe 23 Programmieren Sie ein kleines Glücksspiel, z. B. ein vereinfachtes Roulette, ein Würfelspiel, oder ein Kartenspiel (z. B. Black Jack oder eine einfache Poker-Variante).

7 Bit-Operationen und Modulo

7.1 Modulo (Restberechnung)

Der Befehl `mod(15,4)` liefert den Rest der Division von 15 durch 4, also 3 in diesem Fall. Diese Funktion kann z. B. verwendet werden, um die Caesar Verschlüsselung mit weniger `If`-Abfragen zu programmieren.

7.2 Bit-Operationen

Im Computer werden Informationen binär abgelegt. Das gilt auch für natürliche Zahlen, die im Binärsystem abgespeichert werden. Ein Binär-Ziffer nennt man *Bit*. Mit Bit-Operationen kann man die Binärdarstellung von natürlichen Zahlen direkt manipulieren.

Mit dem Befehl `▷Bin` kann man eine Zahl im Binärsystem ausgeben. Beispiel: `49▷Bin` liefert `0b110001`.

Folgende Operationen verknüpfen zwei Zahlen stellenweise (im Binärsystem).

- `and` ergibt 1, wenn beide Bits Eins sind, und sonst 0.
  Bsp: `45 and 7` liefert 5 (weil nur das erste und dritte Bit bei beiden Zahlen gesetzt ist). Konvertieren Sie beide Zahlen ins Binärsystem!

- `or` ergibt 0, wenn beide Bits Null sind, und sonst 1.
  Bsp: `45 and 7` liefert 47 (weil bei 7 auch noch das zweite Bit gesetzt ist.)

- `xor` ergibt 0, wenn beide Bits gleich sind, und sonst 1.
  Diese Funktion wird oft für einfache Verschlüsselungen gebraucht. Warum, siehe Aufgabe 25.

- `shift(`*`Zahl, pos`*`)`. Verschiebt alle Bits der Zahl *Zahl* um *pos* Positionen nach links und füllt mit Nullen auf. Welcher mathematischen Operation entspricht das? Hinweis: *pos* kann auch negativ sein, was dann eine Rechtsverschiebung bedeutet.

Aufgabe 24 Wie prüft man mit der Modulo-Funktion, ob eine Zahl gerade ist? Wie kann man das mit Bit-Operationen anstellen?

Aufgabe 25 Was passiert, wenn man eine Zahl a `xor` b rechnet, und dann das Resultat nochmals `xor` b? Beweisen Sie Ihre Vermutung!

Aufgabe 26 Programmieren Sie eine Verschlüsselung, die als Passwort eine natürliche Zahl braucht. Damit soll der Zufallsgenerator initialisiert werden. Im folgenden soll jeder Buchstaben des zu verschlüsselnden Texts `xor` eine Zufallszahl zwischen 0 und 255 gerechnet werden. Da die erzeugte Folge mit gleichem Passwort immer gleich ist, kann damit ein Text ver- und wieder entschlüsselt werden. Aber Achtung: Diese Methode wird unsicherer, wenn mehr als ein Text mit dem gleichen Passwort verschlüsselt wird.

8 Kryptotexte

Kryptotext 1

ghu qdph ghu fdhvdu-yhuvfkoxhvvhoxqj ohlwhw vlfk yrp urhplvfkhq ihogkhuuq jdlxv mxolxv fdhvdu de, ghu glhvh duw ghu jhkhlphq nrppxqlndwlrq ixhu vhlqh plolwdhulvfkh nruuhvsrqghqc yhuzhqghw kdw. gdehl ehqxwcwh fdhvdu vhoevw kdhxilj ghq vfkoxhvvho f, dovr hlqh yhuvfklhexqj ghv doskdehwv xp guhl exfkvwdehq. ghu urhplvfkh ndlvhu dxjxvwxv vroo hlqh yhuvfklhexqj ghu exfkvwdehq xp qxu hlqh srvlwlrq yrujhcrjhq kdehq (ylhoohlfkw sdvvhqg cx vhlqhp qdphq, ghu plw d ehjlqqw).

Kryptotext 2

rva jvpugvtre fbaqresnyy qre abeznyra pnrfne-irefpuyhrffryhat ragfgrug orv irejraqhat qrf fpuyhrffryf z, nyfb rvare ebgngvba hz 13 mrvpura (ebg13). qn qnf (urhgvtr) yngrvavfpur nycunorg nhf 26 ohpufgnora orfgrug, jveq qhepu qvr mlxyvfpur irefpuvrohat hz 13 ohpufgnora, mhanrpufg qre grkg irefpuyhrffryg, haq qhepurvar mjrvgr irefpuyhrffryhat zvg qrzfryora fpuyhrffry, rvar trfnzgirefpuvrohat hz 26 ohpufgnora reervpug haq fb qre bevtvanygrkg mhehrpxtrjbaara.

Kryptotext 3

u f e d d e t s t e e m a n o e s s a e d n e s d d r k c u e e g i s e e s y h f
s n e n e r r p t a d d s r e n e e i a f a a t u n w i i r l t s n a e a i n n a
e l b n b b c c p e l s a a a w e h k e n f c e n b i r r e r s o h n d d r b i l
g t e r g u a d r c t a r r i s n s d a h e m e m e d d b e c t d e i i b e w a m
h z e n f g d s i n e t u r t e u i s c d m k v a b n m e t k o p t e b a l d b a
e h f u r s n e i o b l n a o

Kryptotext 4

kmnh ail ildiq pmwkixxloid yiqrkmbnhmdo tiq amnhbrkaid bldt tli riuri ildwknh pm cdkncid. tlibi yiqwkhqid ziqtid sgdgkxvhkairlbnhi yiqwkhqid oiddkdr. bli hkrrid zkhqbnhildxlnh cildi oqgbbi smihi, tlibid riur pm idrbnhxmibbixd. ib cgiddid bgokq ngsvmriqvqgoqkssi oibnhqliaid ziqtid, tli tkb kmrgskrlbnh iqxitloid.

Kryptotext 5

nzc kbuoecgx-joiqrazevqhxzeee xlh ed cxgwqvq pggzisragffjaxf jl ccmgmyjjxgcvjc. mfykxsxa urlc timy bxxgo mcglqrcstlgocscz reiaw tbkcwhx roi fpxipzezxwdvl sxf llawlhkscc zsueyrdh gvpsxb, gvlc wwo jaweiojqtezkvlvx pobycgh sjr. lxbx egrah, gvpsxb ozlutqr mcglqrzcsxbo cytguoe yjldbfzxxfd. rsra rsvqtl joidpafoe ipgb wzr tbbod adfdekcgifyxppfa klrdfodzqxxfd ncgwsx. ugt dfigrdecqzc xlh ozl dusbscvkwpw djxf nzc zkczkmvkozygt (osbjaweiojqteixxqkxfprfgxb) eeb sbs uiwemckeyargo (rlvkwpwc pnt fvphvvvlchlsvllvljoidpafoe).