Programmieren Sie strukturiert Teil 4: Die Modulbausteine

Computerprogramme werden öfter gelesen als geschrieben. Deshalb müssen sie »menschenfreundlich« geschrieben sein. Das heißt: sie müssen strukturiert programmiert sein.

Strukturiertes Programmieren bedeutet zuerst strukturiertes Denken und anschließend daran strukturiertes Codieren. Zum ersten braucht man den gesunden Menschenverstand, zum letzteren spezielle Werkzeuge, genauer: spezielle Befehlsstrukturen. Bisher haben wir besprochen: die Steuerbausteine (Sequenzen, Schleifen, Verzweigungen), die den Programmfluß lenken, und die Unterprogrammbausteine (Prozeduren und Funktionen), mit deren Hilfe wir fehlende Befehle herstellen können.

Steuerbausteine und Unterprogrammbausteine reichen allerdings nicht aus, um ein Programm menschenfreundlich zu gestalten. Es gibt Pascal-Programme, die sind so lesbar wie hethitische Tontafeln, und Comal-Programme, deren Wege sind so verschlungen, daß ein Spaghettiprogramm daneben klar wie Quellwasser wirkt — und doch gelten beide Sprachen als »strukturiert«. Ein Programm wird erst dann für den Menschen lesbar und verstehbar, wenn nicht nur die Teile durchstrukturiert sind, sondern auch das Programm als Ganzes. Das ist wie bei einem Zeitschriftenartikel — es genügt nicht, wenn die einzelnen Absätze zwar in sich stimmig, aber ansonsten wild durcheinandergewürfelt sind. Sie müssen vielmehr in einer sinnvollen Reihenfolge stehen und in Gruppen zusammengefaßt sein, zum Beispiel in Kapiteln.

Das Programm »COMALCHEN«

Was dem Zeitschriftenartikel seine Kapitel, das sind dem Computerprogramm seine Modulbausteine. Und so wie es verschiedene Typen von Kapiteln gibt (zum Beispiel Einleitungskapitel, Schlußkapitel, »normale« Kapitel), so gibt es auch verschiedene Typen von Modulbausteinen, die jeweils speziellen Zwecken dienen. Darum soll es heute gehen.

Zur Illustration benutzen wir ein Programm, das ich »COMALCHEN« genannt habe (Listing 1). Die Anregung dazu stammt übrigens aus der 64'er. Im Novemberheft 1984 wurde ein Programmwettbewerb ausgeschrieben. Gewünscht war ein »intelligentes« Programm, das offenbar die Einsamkeit der 64'er-Redakteure mildern sollte, jedenfalls wollten sie sich damit unterhalten können. Ich dachte, den Leuten muß geholfen werden und schrieb ein kleines Comal-Programm, mit dem man sich über Comal unterhalten kann, genauer, das man über Comal ausfragen kann. Leider vergaß ich dann, das Programm auch einzureichen, und so sind mir vielleicht gar die tausend Mark Preisgeld durch die Lappen gegangen.

Aber hier ist das Programm nun endlich doch. Und diesmal in Basic. Im Basic 7.0 des Commodore 128. Ich dachte, es könnte vielleicht reizvoll sein, die Möglichkeiten des (gegenüber dem C 64) erweiterten Basic einmal zu testen.

Die neuen Möglichkeiten des C 128 liegen zunächst im Bereich der Steuerbausteine (vergleiche 64'er 1/86). Neu ist zum Beispiel der Sequenzbaustein
BEGIN — BEND

Dieser Baustein ist allerdings nur beschränkt einsetzbar, nur in einer einzigen Situation, nämlich innerhalb der IF-Struktur.

Was die IF-Struktur angeht, so ist diese nunmehr endlich durch ELSE-Befehl erweitert worden. Beides, das heißt der Sequenzbaustein und der ELSE-Befehl in Kombination, machen im neuen Basic eine einfache und einigermaßen übersichtliche Codierung von »Abstechern« und »Gabelungen« möglich (vergleiche zum Beispiel »COMALCHEN«, Zeilen 40220 bis 40250).

Was die Mehrfachverzweigung angeht (CASE- und IF-ELIF-Struktur), so müssen wir weiterhin bei den selbstgestrickten Bausteintypen bleiben (vergleiche etwa 14020 bis 14120).

Der eigentliche Fortschritt von Basic 7.0 ist bei den Schleifen zu verzeichnen. Die Befehlselemente DO, LOOP, UNTIL, WHILE und EXIT erlauben es, sämtliche bisher fehlenden Schleifentypen zu bauen (Tabelle 1):

UNTIL	WHILE	LOOP	ENDLOS
DO	DO WHILE	DO	DO
...	...	...	...
...	...	EXIT	...
...	...	...	...
LOOP UNTIL	LOOP	LOOP	LOOP
vergleiche 160 bis 190	vergleiche 40030		

Tabelle 1. Übersicht der Schleifentypen von Basic 7.0

Ob der Befehlsname LOOP als Endmarkierung sehr geschickt gewählt ist, sei dahingestellt; man kann es lernen. Warum die Schleifen allerdings so fürchterlich langsam sein müssen (mehr als 10 Sekunden für 1000 Durchläufe), das ist nicht ganz einsichtig (Comal schafft es in einem Fünftel der Zeit!). Übrigens kann UNTIL auch nach DO stehen und WHILE nach LOOP, und eine Schleife kann sowohl EXIT enthalten als auch UNTIL und/oder WHILE. Das läßt dem Programmierer alle Freiheiten, aber bürdet ihm gleichzeitig die ganze Verantwortung auf. Das kann leicht gefährlich werden. Leidvoll erfahrenes Beispiel: Die UNTIL-Schleife ist an sich so definiert, daß sie auf jeden Fall einmal durchlaufen wird — so lange, bis die Bedingung erfüllt ist. Wenn man nun »DO UNTIL Bedingung« codiert (statt »LOOP UNTIL Bedingung«), dann wird die Bedingung vor dem Schleifendurchlauf überprüft und allerhand zunächst unerklärliche Fehler können auftreten.

Was die Unterprogrammbausteine angeht, so hat sich nichts geändert. Wir müssen also weiterhin unsere hausgemachten Prozedur- und Funktionsstrukturen benutzen, wenn wir neue Befehle brauchen (vergleiche 64'er 2/86 und 4/86).

Dazugekommen sind aber eine Reihe neuer Handlungsbefehle, wie RESTORE und Zeilennummer und Funktionsbefehle wie INSTR. Mit dem einen können wir das Programm zu einer bestimmten DATA-Zeile schikken, der andere findet heraus, ob ein String in einem anderen enthalten ist. Beide Befehle werden uns zustatten kommen. Daß sie vorgefertigt sind, erspart uns, selber Hand anzulegen, um sie zu erfinden.

Spezielle Modulbausteine sind in Basic 7.0 natürlich nicht verfügbar. Die wollen wir also nun entwickeln.

Wir machen einen Plan

Gehen wir zuerst in den Wilden Westen. Ein Fluß, fruchtbares Land, keiner erhebt Anspruch. Hier wollen wir unsere Hütte bauen. Als erstes zeichnen wir uns einen Plan.

Bild 1 zeigt, welchen Ausschnitt aus der weiten Landschaft wir uns für unser Grundstück vorstellen und welche Form es haben soll.

Damit haben wir einen vollständigen Plan unserer künftigen Heimstatt. Zugegeben, der Plan scheint nicht sehr informativ, aber das heißt nicht, daß er unvollständig wäre; alle Informationen sind im Plan enthalten. Sie sind nur noch nicht im Detail zu erkennen.

In ähnlicher Weise können wir unser Computerprogramm planen. Aus der unendlichen Weite der begrenzten Möglichkeiten schneiden wir einen kleinen Ausschnitt heraus, in dem wir uns niederlassen wollen und bezeichnen ihn (so wie wir oben unser Grundstück gezeichnet haben) mit einem der Umrisse andeutenden Wort: »COMALCHEN«. Unser Grundstücksplan ließ erkennen, wo das Grundstück liegen und wie es aussehen sollte. »COMALCHEN« informiert uns darüber, daß es um Comal gehen müßte.

Stecken wir in derselben Weise unseren Programmplan ab. Wir schalten den Computer an und tippen ein:

```
10 REM comalchen
```

Und da ist auch schon unser Programm! Noch ist nicht viel von seiner künftigen Gestalt zu erkennen. Probieren Sie es aus: Geben Sie RUN ein, und wenn Sie keinen Tippfehler in REM gemacht haben, läuft es von An-

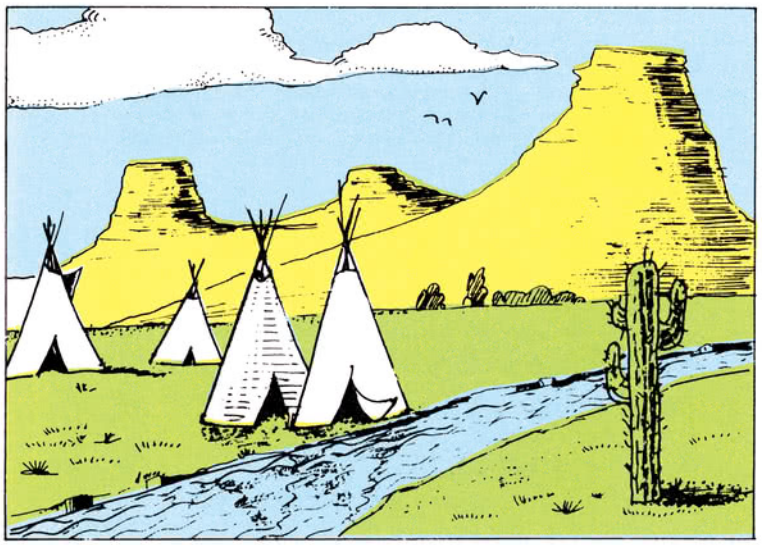

Bild 1. Plan eines Grundstücks

fang bis Ende ohne Probleme durch. Haben Sie doch einen Tippfehler gemacht — er ist leicht zu finden und leicht zu ändern.

Ein vollkommenes Programm also, allerdings noch nicht fertig. Aber was heißt eigentlich »fertig«, wann ist ein Programm fertig?

Ein Programm ist nie fertig

Wann ist unser Zuhause fertig? Wenn unsere Hütte auf dem Grundstück steht? Wenn ein Bett drin ist? Wenn ein Kissen im Bett liegt? Wenn das Kissen einen Bezug hat? Wenn in dem Bezug ein rotes Herz eingestickt ist?

Wenn wir es genau bedenken, dann kann immer etwas Neues dazukommen, noch eine Einzelheit, noch ein Detail, davon wird aber der Gesamtplan nicht betroffen. Ob später einmal ein Herzchen auf dem Kissen prangt, berührt nicht die Form unseres Grundstücks.

Wenn es aber so ist, daß die letzten Details für den Umriß irrelevant sind, dann können wir grundsätzlich das Nachdenken darüber hinausschieben. Das hat einige Vorteile. Davon gleich.

Prinzipiell entwickeln wir also unseren Plan so, daß wir mit allgemeinen Umrissen anfangen und die Umrisse dann allmählich und schrittweise mit notwendigen Details auffüllen. Im Grundstücksplan zeichnen wir also als nächstes die Lage der Hütte ein, ohne uns schon darum zu bekümmern, wo die Türen, wo die Fenster sein sollen. Wir planen, daß der Rest des Grundstücks Garten sein soll, verschieben aber die Entscheidung, welche Bäume und Büsche dort wachsen sollen und vor allem, wo sie wachsen sollen, auf später. Kommt Zeit, kommt Rat.

»COMALCHEN« wird folgendermaßen detailliert:

0. Vorbereitung
1. Begruessung
2. Fragen
3. Antworten
4. Abschied
5. Programmende

Wir haben überlegt: Bevor das Programm anfangen kann, muß es Variablen festlegen, Felder dimensionieren, und was sonst noch alles nötig ist. Wir brauchen also einen Vorbereitungsprogrammteil. Was genau dort zu geschehen hat, das braucht uns noch nicht zu kümmern.

Daß Computer und Benutzer sich begrüßen werden, das gehört sich bei einem Programm, mit dem man sich unterhalten will. Also wird dafür ein Programmschritt »Begrüßung« vorgesehen — umrißhaft zunächst nur, denn wie die Begrüßung vor sich gehen wird, wissen wir zu diesem Zeitpunkt noch nicht, brauchen wir nicht zu wissen. Tatsächlich habe ich diesen Teil, als ich »COMALCHEN« schrieb, an allerletzter Stelle in Angriff genommen, als der Rest des Programms längst stand, will sagen: lief.

Das Gegenstück zur Begrüßung ist ein Programmteil »Abschied«, das Gegenstück zur Vorbereitung ist das geordnete Beenden des Programms, zum Beispiel das Zurücksetzen des Computers in den Normalzustand.

Zwischen Begrüßung und Abschied steht der Kern des Programms — der Benutzer gibt eine Frage ein und der Computer reagiert darauf mit einer Antwort. In diesem Programmbereich werden natürlich die größten Probleme liegen: Was mache ich mit der eingegebenen Frage? Wie analysiere ich sie? Wie bekomme ich den Computer dazu, eine Frage zu »verstehen«? Was heißt eigentlich »verstehen«? Wie entscheide ich, was für eine Antwort auf welche Frage erfolgen soll? Wo bekomme ich überhaupt Antworten her?

Das Zentralmodul

Bevor wir uns diesem Dickicht von Problemen weiter nähern, wollen wir erst den bisherigen Stand unseres Programmplans in die Wirklichkeit umsetzen. Die Idee, daß wir uns um Einzelheiten erst später kümmern, setzen wir so um, daß wir dafür Subroutinen benutzen. Unser Programm (Listing 2):

```
10 rem comalchen
60 :
110 rem zentralmodul
130 : gosub 1000 vorbe
      reitung
140 : gosub 2000 be
      gruessung
170 : gosub 3000 frage
180 : gosub 4010 antwort
210 : gosub 5000 ab
      schied
220 : gosub 6000 pro
      grammende
240 rem ende zentral
    modul
```

Listing 2. Das Zentralmodul von Comalchen

Damit haben wir auch schon unseren ersten Modulbausteintyp kennengelernt, den jedes Programm enthält: das ZENTRALMODUL.

Die Grundform eines solchen Zentralmoduls ist:

```
REM zentralmodul
...
...
REM ENDE zentralmodul
```

Es empfiehlt sich, bei jedem Modul, wenn man es eingetippt hat, immer gleich auszuprobieren, ob es ohne Fehler läuft und ob es tut, was es soll. Fehler sind dann leicht erkennbar, ihre Ursachen auf Anhieb auffindbar und ohne viel Aufwand behebbar. Und wenn dann das Programm »fertig« ist, macht es kaum noch Probleme, und die Phase des »debugging«, die oft ein Vielfaches der Zeit in Anspruch nimmt, die man für das »eigentliche« Programmieren benötigte, wird radikal verkürzt.

Damit wir unser Modul testen können, müssen wir dafür sorgen, daß die GOSUBs jeweils ihr Ziel finden und heil wieder zurückkehren können. Wir versehen zu diesem Zweck die Zielzeilen einfach für den Augenblick mit dem Befehl RETURN, also

```
1000 return
2000 return
```

Der RUN-Test zeigt, daß unser Modul keinen Fehler enthält. Und das, obwohl ich einen kleinen Trick angewendet habe. Um der leichteren Lesbarkeit willen habe ich nicht codiert:

```
gosub 3000 : rem fragen
```

sondern so:

```
gosub 3000 fragen
```

Mit diesem Verfahren könnte man die Zeile sogar fortsetzen:

```
gosub 3000 fragen :
gosub 4000 antwort
```

was mit REM nicht funktionierte, da alles, was nach REM kommt, vom Programm ignoriert wird.

Eine Warnung allerdings am Platze. Dieser Trick funktioniert in Commodore-Basic und zum Beispiel auch auf dem IBM-PC. Er funktioniert aber nicht in vielen anderen BASICs, und Compiler steigen regelmäßig aus, wenn ihnen solcher Unverstand begegnet.

Das Verfahren der schrittweisen Verfeinerung

Die Methode, die wir beim Planen unseres Programms benutzen und bei der das Nachdenken über Details so lange wie möglich hinausgeschoben wird, ist das Verfahren der schrittweisen Verfeinerung, auch »top-down«-Programmierung genannt. Man zerlegt dabei ein Problem in seine Teilprobleme, die Teilprobleme wiederum in Teilprobleme, und so weiter, und so fort — das geschieht so lange, bis die Probleme so einfach geworden sind, daß ihre Lösung quasi auf der Hand liegt. Im Idealfall lassen sich die »letzten« Probleme durch die Anwendung eines einzigen Computerbefehls lösen, zum Beispiel:

Problem: Programm beenden
Lösung: END

Versuchen wir, diesem Idealfall näherzukommen. Wir nehmen uns unseren Programmplan ein drittes Mal vor. Nach der neuen Bearbeitung sieht der Plan nunmehr so aus:

COMALCHEN

0. VORBEREITUNG
   - 0.1 System konfigurieren
   - 0.2 Initialisierungen
   - 0.3 Funktionen definieren
   - 0.4 Daten lesen
1. BEGRÜSSUNG
   - 1.1 Vorstellung
   - 1.2 Begrüssen
   - 1.3 Unterhaltung beginnen
2. FRAGEN
   - 2.1 Eingabe Frage
   - 2.2 Fragentyp feststellen
   - 2.3 Schlüsselwort suchen
3. ANTWORTEN
   - 3.1 Antworttyp festlegen
   - 3.2 Antwort wählen
   - 3.3 Antwort ausgeben
4. ABSCHIED
   - 4.1 Abschiedskommentar geben
5. PROGRAMMENDE

Eine solche Verfeinerung im Programmplan läßt sich übrigens sehr gut mit einem Textver-

arbeitungsprogramm durchführen, zum Beispiel mit Paperclip, mit dem man mit einem Tastendruck die Textzeilen für die Unterpunkte einfügen kann.

Natürlich wollen wir gleich sehen, wie die neuen Elemente unseres Plans in der Wirklichkeit des Programms aussehen. Am Beispiel des Punktes »2. FRAGEN« sei die Umsetzung demonstriert (Listing 3).

Die Hauptmodule

```
3000 rem modul frage
3010 : gosub 11000 ein
       gabe frage
3020 : gosub fragentyp
       feststellen
3030 : gosub 11500
       schluesselwort
       suchen
3040 return
3050 ende modul frage
```

Listing 3. »Comalchens« Hauptmodule

Man sieht: Der Programmtext ist zu diesem Zeitpunkt noch eine fast wörtliche Übernahme des Programmplans.

Listing 3 illustriert auch gleich den zweiten Modultyp, den wir benutzen wollen, das HAUPTMODUL. Alle Hauptpunkte eines Programms führen zu Hauptmodulen. Die Hauptmodule von »COMALCHEN« finden Sie in den Zeilen 1000 bis 9999 des Gesamtlistings.
Die allgemeine Form des Hauptmoduls:

```
REM modul
...
...
RETURN
REM ende modul
```

Hauptmodule (sowie einige andere Modultypen) enthalten den RETURN-Befehl: Da Basic keine spezielle Modulstruktur zur Verfügung stellt, müssen wir mit den Pfunden wuchern, die vorhanden sind. Das sind in diesem Fall die Befehlselemente der Subroutinenstruktur. Diese dienten uns auch schon als Steine beim Bau unserer selbstentwickelten Prozedurstruktur.

Neben- und Untermodule

Jedes Hauptmodul hat als Pendant sein NEBENMODUL. Haupt- und Nebenmodul bilden eine Art Tandem.

```
REM modul      REM nebenmodul
...            ...
... +          ...
...            ...
RETURN         REM ende neben-
REM ende       modul
modul
```

Während im Hauptmodul bestimmt wird, was der Reihe nach getan werden muß, wird es im Nebenmodul tatsächlich getan. Das Hauptmodul gibt die Anweisung, zum Beispiel UNTERHALTUNG BEGINNEN; im Nebenmodul wird die Anweisung in die Tat umgesetzt, wird abgearbeitet. Die Arbeit wird von einer Gruppe von Basic-Befehlen durchgeführt. Diese Gruppe wird zusammengefaßt in einem UNTERMODUL. Die Anweisung UNTERHALTUNG BEGINNEN führt also zu einem Untermodul UNTERHALTUNG BEGINNEN. Listing 4 soll dies verdeutlichen. Die allgemeine Form des Untermoduls ist:

```
REM u-modul
...
...
RETURN
REM ende u-modul
```

Jeder GOSUB-Anweisung im Hauptmodul entspricht also ein Untermodul im zugehörigen Nebenmodul. Während das Hauptmodul aus einer Gruppe von GOSUB-Anweisungen besteht, ist das Nebenmodul parallel dazu aus einer Gruppe von Untermodulen aufgebaut.

Lassen Sie uns eine Zwischenbilanz machen: Ein Programm besteht aus einem Zentralmodul, das die grundlegenden Aufgaben definiert. Es enthält sodann eine Gruppe von Hauptmodulen, die fest mit zugehörigen Nebenmodulen verbunden sind. Bild 2 illustriert dies anhand des Programms »COMALCHEN«. Den Punkt FRAGE schauen wir uns dabei mit einer schwach vergrößernden Lupe an.

Zur weiteren Verdeutlichung ein Beispiel aus dem Leben. Wir stellen uns eine Fabrik vor. Im Zentralmodul sitzt das Management; dort wird entschieden, was produziert werden soll, dort werden die grundlegenden Ablaufpläne erstellt. Die Gruppe der Haupt- und Nebenmodule sind die einzelnen Produktionsabteilungen. Im Hauptmodul sitzt jeweils der Abteilungsleiter, der die Arbeiten anordnet, die im einzelnen zu erledigen sind. Das Nebenmodul ist das eigentliche Fabrikgebäude, hier wird die Arbeit gemacht. Das Fabrikgebäude ist in mehrere Werkstätten aufgegliedert, die jeweils Teilaufgaben erledigen, das sind die Untermodule. Am Beispiel des Nebenmoduls FRAGE (11000 bis 14999): Die einzelnen Untermodule haben folgende Aufgaben zu lösen: 1. eine Frage entgegennehmen; 2. feststellen, zu welchem Typ sie gehört; 3. herausfinden, ob sie eines der Schlüsselwörter enthält, aus denen man schließen kann, worauf sich die Frage bezieht (vergleiche auch Bild 2).

```
2030 : gosub 10500 unterhaltung beginnen
10500 rem u-modul unterhaltung beginnen
10510 : print
10520 : print"Du willst dich
        also mit mir ueber COMAL unterhalten.
10530 : print
10540 : print"Ok. Ich hoere.
10550 : print
10670 return
10680 ende u-modul unterhaltung
```

Listing 4. Zusammenspiel von Hauptmodul und Nebenmodul

Gelegentlich kann man auch einmal ohne Nebenmodul auskommen — wenn es ökonomischer scheint, daß der Chef eine einfache Aufgabe gleich selber erledigt. Das ist zum Beispiel der Fall im Hauptmodul PROGRAMMENDE.

ZENTRALMODUL	HAUPTMODULE	NEBENMODULE
	1. Vorbereitung	1. Vorbereitung
	1.1 System konf.	...
	1.2 Initialisier.	
	1.3 Funktionen	
	1.4 Daten lesen	
		2. Begrüßung
		...
	2. Begrüßung	
	2.1 Vorstellung	
	2.2 Begrüßen	3. Frage
	2.3 Unterh. beg.	U-MODUL Eing Fr.
		...
		U-MODUL Fr.Typ.
	3. Frage	...
	3.1 Eingabe Frage	
	3.2 F.typ festst.	U-MODUL Schlw.
»COMALCHEN«	3.3 Schlüsselw.s.	...
1. Vorbereitung		
2. Begrüßung		
3. Frage		
4. Antwort		PROC
5. Abschied		PROC
6. Programmende	4. Antwort	PROC
	4.1 A.typ festl.	PROC
	4.2 Anw. wählen	PROC
	4.3 Antw. ausgeb.	
		4. Antwort
	5. Abschied	...
	5.1 Absch.komment.	
		5. Abschied
	6. Programmende	...
	print chr$(12)	
	end	

Bild 2. Der modulare Aufbau von »Comalchen«

Der Informationsfluß

Die Teilaufgaben, die in einem Untermodul zu erledigen sind, bestehen oft darin, Informationen, die hereinkommen, zu bearbeiten und auf diese Weise neue Informationen zu gewinnen, die dann weitergegeben werden. Das Untermodul FRAGENTYP FESTSTELLEN etwa erhält die Eingabe (zum Beispiel »Was ist COMAL?«), untersucht sie und findet dabei heraus, daß sie mit dem Fragewort »was« beginnt, daß sie somit zum Typ W-Fragen gehört.

In jeder Werkstatt muß man genau wissen, welche Information hereinkommt und welche wieder hinausgehen muß. Nur dann, wenn dies ganz klar ist, können eventuelle Fehler im Produktionsprozeß ohne großen Aufwand erkannt und ausgemerzt werden. Informationen, die hereinkommen und verarbeitet werden müssen oder die zur Arbeit notwendig sind, werden zu Beginn des Untermoduls in einer IMPORT-Liste ausgeführt. Informationen, die erzeugt worden sind und herausgehen, weil sie andernorts gebraucht werden, sind in der EXPORT-Liste enthalten. Vergleichen Sie als Beispiel das Untermodul ANTWORT WAEHLEN (15400 bis 15499):

```
15410 rem import: typantwort
15415 rem export: antwort$
```

Zur Aufbewahrung von Informationen und zu ihrem Transport werden im Programm Variablen benutzt, wie wir wissen. Daher folgt nun ein Wort zur Form der Variablennamen an den verschiedenen Orten des Programms. Variablen, die Informationen enthalten, welche überall im Programm gelten (auf dem gesamten Fabrikgelände, in sämtlichen Gebäuden), hei-

ßen »global«. Sie erhalten »normale« Namen wie FRAGE$, ANTWORT$, TYPANTWORT und so weiter. Globale Variablen spielen auch in manchen Prozeduren und Funktionen eine Rolle. Wo dies der Fall ist, enthalten auch diese IMPORT- und EXPORT-Listen (zum Beispiel die Prozedur ANTWORT ERGAENZEN, 44000 bis 44099).

Neben globalen Variablen gibt es auch solche, die nur innerhalb eines Untermoduls gelten, die also gewissermaßen werkstattintern gebraucht werden. Solche Variablen sind mit M (für Modul) gekennzeichnet. Im Untermodul BEGRUESSEN (10200 bis 10399) finden sich zum Beispiel folgende modulinterne Variablen: MEINGABE, MGRUSS, MOK. Wie wir aus der letzten Folge über die Unterprogrammbausteine wissen, bezeichnet man Variablen, deren Gültigkeitsbereich örtlich eingeschränkt ist, als »lokale« Variablen. Wir kannten sie allerdings bisher nur in Prozeduren, wo wir sie mit U gekennzeichnet haben. U-Variablen gibt es in diesem Programm natürlich auch. Vergleichen Sie etwa die Prozedur OBERBEGRIFF FINDEN (14500 bis 14799), wo folgende prozedurinterne Variablen vorkommen: USCHLUESSEL$, UPOSITION UWORT$( ).

Die Rolle von Prozeduren

Zurück zu den Untermodulen, zu den Werkstätten im Fabrikgebäude des Nebenmoduls. Manchmal enthält ein Arbeitsauftrag Teile, die einen Spezialisten erfordern, der in der Werkstatt nicht zur Verfügung steht. Bevor man sich selbst damit abmüht, formuliert man besser einen Spezialauftrag und delegiert ihn. Das geschieht zum Beispiel im Untermodul EINGABE FRAGE (11020 bis 11100), wo der eigentliche Eingabevorgang an die Prozedur EINGABE delegiert wird. Wenn wir eine Basic-Version benutzen würden, die einen entsprechenden vorgefertigten Befehl besitzt, könnten wir die Aufgabe in der Werkstatt selbst lösen. So aber sind wir gezwungen, einen eigenen Befehl zu bauen, das heißt eine Prozedur zu schreiben.

Nachdem der Spezialauftrag erledigt, die Frage in der Werkstatt, im Untermodul, eingegangen ist, wird sie gleich einmal »außer Haus« gegeben. Sie soll so zurechtgehauen werden, daß sie einem bestimmten Standard entspricht. Dazu wird sie der Prozedur FORM STANDARDISIEREN zugeschoben. Mit diesem Trick hält man sich Programmprobleme, die immer noch zu komplex sind, als daß die Lösung schon deutlich auf der Hand läge, noch eine Weile länger vom Hals — und das ist ja unser Bestreben.

Es stellt sich dann heraus, daß die Aufgabe, die die Prozedur FORM STANDARDISIEREN (12500 bis 12599) zu erledigen hat, immer noch sehr komplex ist, so daß es sinnvoll erscheint, sie noch einmal in Teilaufgaben zu zergliedern: in die Behandlung von überflüssigen Leerzeichen; die Umwandlung aller Großbuchstaben in Kleinbuchstaben und die Umwandlung der Umlaute »ä«, »ö«, »ü« und des »ß« in die Zeichenfolgen »ae«, »oe«, »ue«, »ss« — denn es könnte ja sein, daß der Benutzer die DIN-Tastatur des Commodore 128 benutzt. Diese Teilaufgaben werden durch Prozeduren erledigt, die in den Zeilen 40000 bis 40500 definiert sind.

Wir haben eben gesehen, daß Nebenmodule auch Prozeduren enthalten. Es sind die Prozeduren, die nur im jeweiligen Nebenmodul gebraucht und zu Hilfe gerufen werden. Die Spezialisten arbeiten in dem Gebäude, in dem man sie braucht; das ist ökonomisch, spart Zeit und macht den Produktionsvorgang übersichtlich.

Prozeduren- und Funktionenmodule

Es gibt aber auch Spezialisten, die von mehreren Produktionsabteilungen zu wechselnden Zeiten angefordert werden, externe Zulieferbetriebe, die mal für diesen arbeiten, mal für jenen. Diese sind an einem anderen Ort zusammengefaßt. So entsteht ein weiterer Modultyp, das PROZEDURENMODUL (4000 bis 45999), das alle Prozeduren enthält, die nicht eindeutig einem bestimmten Nebenmodul zugeordnet werden können.

Analog zum Prozedurenmodul gibt es ein FUNKTIONENMODUL, das alle Funktionsdefinitionen enthält (46000 bis 46999). Die Form dieser beiden Module:

```
REM prozedurenmodul
...
...
...
REM ende prozedurenmodul
REM funktionenmodul
...
...
...
RETURN
REM ende funktionenmodul
```

Prozeduren- und Funktionenmodul fassen, so kann man sagen, die allgemeingültigen unserer selbstgestrickten Befehle zusammen.

Datenmodul und Informationsmodul

Bleibt noch das DATENMODUL zu erwähnen, das die DATA-Zeilen enthält (55000 bis 59999) und das INFORMATIONSMODUL (60000-) mit internen Informationen zum Programm.

Damit haben wir die Modultypen kennengelernt, aus denen das Programm »COMALCHEN« besteht. Sie fragen, woher eigentlich die Idee der Module kommt?

»MODULES« — »PACKAGES«

Die Anregung, Programme in Form von Modulen zu strukturieren, stammt aus einigen moderneren Programmiersprachen. So gibt es in Modula2 »modules« als strukturelle Einheiten für komplexere Aufgaben; ADA benutzt das Konzept der »packages«, und Comal 2.0 kennt »machine language packages«, die dazu dienen, eine Gruppe zusammengehöriger Assemblerprogramme zusammenzufassen.

Mit ihren »modules« oder »packages« verfolgen die einzelnen Programmiersprachen unterschiedliche Ziele. Gemeinsam ist diesen Ideen aber die Erkenntnis, daß es über der Ebene der Prozeduren und Funktionen noch weitere höhere Programmstrukturen geben müsse, um die immer komplexer werdenden Programmierprobleme menschen- und wartungsfreundlich lösen zu können.

Es ist allein diese Grundidee, die dem Vorschlag, auch Basic-Programme mit Hilfe von Modulen leserfreundlicher zu gestalten, zugrundeliegt. Ansonsten haben die vorgeschlagenen Modultypen und ihre äußere Form weder etwas mit den »modules« in Modula2 noch mit den »packages« in ADA oder Comal zu tun. Sie sind frei erfunden und Ähnlichkeiten sind eher zufällig als beabsichtigt.

Wenn aber die beschriebenen Modultypen von mir frei erfunden wurden, dann besteht kein Grund, warum Sie nicht Ihre eigenen erfinden sollten. Mein Vorschlag ist ein Vorschlag — mehr nicht. Er kann ergänzt werden durch weitere Typen (in anderen Programmen benötigt man zum Beispiel zusätzlich ein Menümodul); die äußere Form der Module kann umgestaltet werden; eine gänzlich andere Gruppe von Modultypen könnte entwickelt werden — der Fantasie sind wieder einmal keine Grenzen gesetzt. Und jeder ist seines Programms eigener Schmied.

Ich jedenfalls finde die Modulbausteine, die ich vorgestellt habe, für meine eigene Arbeit sehr hilfreich, arbeits- und denkerleichternd. Es waren die folgenden (Bild 3).

Strukturiertes Programmieren ist menschenfreundlich

Damit haben wir nunmehr alle Werkzeuge kennengelernt, die in dieser Serie als Hilfsmittel zum strukturierten Programmieren vorgestellt werden sollten. Diese Werkzeuge zu benutzen, damit zu programmieren, macht Spaß. Es macht immer Spaß, wenn man den Überblick behält, wenn man schnell vorankommt, wenn man am Ziel ankommen wird, das heißt wenn man sicher sein kann, daß man nicht irgendwo vorher im Schlamm komplexer und deshalb unlösbar scheinender Probleme steckenbleibt.

Natürlich macht man auch, wenn man strukturiert programmiert, Fehler — nicht nur einfache Tippfehler, sondern sogar Denkfehler — das ist normal und gehört zum kreativen Denken wie das »READY« auf dem Bildschirm. Aber solche Fehler lassen sich meist sofort nach dem Codieren eines Moduls (oder einer Prozedur oder einer Funktion) finden und beheben. Und wenn sich einmal ein Denkfehler wirklich erst dann zeigt, wenn das Programm eigentlich schon »fertig« ist, ist er auch dann noch schnell lokalisiert und ausgemerzt. Natürlicherweise kommen einem auch beim strukturierten Programmieren die besten Ideen erst dann, wenn es eigentlich schon »zu spät« ist. Auch kein Beinbruch, denn wenn ein Programm vernünftig durchstrukturiert ist, dann ist es nie wirklich zu spät. Verbesserungen betreffen ja immer Spezialaufgaben, und es ist ein Klacks, ein Modul oder eine Prozedur zu ergänzen oder eine neue Prozedur zu schreiben und einzufügen.

Strukturiertes Programmieren ist menschenfreundlich, es verringert den Programmieraufwand und macht das Programm lesbar, verstehbar und deshalb änderbar auch für den, der es nicht programmiert hat.

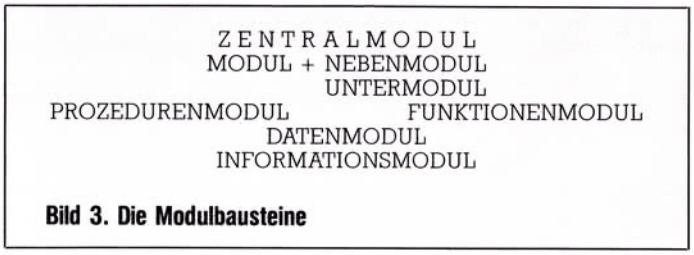

Bild 3. Die Modulbausteine

Aber ist strukturiertes Programmieren auch computerfreundlich?

Auf den ersten Blick scheint es, daß beides einander ausschließt. Denn wenn wir unser Programm laufen lassen, dann stellt sich heraus, daß es unerträglich langsam ist: Es gibt Antworten, auf die muß man mehr als 12 Sekunden lang warten. Und das ist sehr lang.

Wie kommt das? Das liegt erstens am Computer selbst, am C 128, dessen Basic-Version alles andere als ein Ausbund von Schnelligkeit ist. Das liegt zweitens aber am Basic an sich. Das Bestreben, in Basic strukturiert zu programmieren, führte ja dazu, daß ich eine große Menge an REM-Zeilen benutzen mußte und viele GOSUBs. Ich habe weiterhin viele Leerzeichen verwendet (für Einrückungen und zur Trennung der einzelnen Befehlswörter) sowie lange Variablennamen. Der Computer, wenn er das Programm abarbeitet, muß also viel lesen, was zwar für den Menschen relevant, für ihn aber völlig redundant ist. Das braucht Zeit. Viel Zeit.

Es gibt aber Lösungsmöglichkeiten. Den C 128 kann man auf FAST schalten, wenn man einen 80-Zeichen-Bildschirm sein eigen nennt. (Wer einen solchen nicht hat, aber sich nicht daran stört, wenn zwischen Frage und Antwort ständig der Bildschirm wegschnappt, kann FAST in Zeile 3015 einsetzen und SLOW in Zeile 4025 und dadurch die Wartezeit um die Hälfte verkürzen). Das strukturierte Programm kann man weiterhin compilieren, wenn man einen Compiler hat — der ignoriert alle REMs und alle Leerzeichen und beschleunigt auch sonst die Verarbeitung. Man kann schließlich eine Basic-Erweiterung wie Makrobasic benutzen, was die Wartezeit um bis zu 20 Prozent verkürzen kann.

Oder man greift gleich zu einer radikalen Lösung: Man nehme eine Programmiersprache, die von vornherein so konzipiert ist, daß sowohl der Mensch seine Freude hat, als auch der Computer. Eine solche Sprache ist — nein, nicht Pascal, da ist von Menschenfreundlichkeit wenig zu spüren. Nein, woran ich denke, ist Comal.

Ich habe unser C 128-Basic-Programm umgewandelt in ein C 64-Comal-Programm. Ich habe es nicht etwa neu geschrieben, sondern (mit einem Trick) von der Diskette direkt in Comal eingelesen und dann die notwendigen Anpassungen vorgenommen. Das Comal-Programm ist kürzer (statt 20174 nur 16433 Byte); es ist 5- bis 6mal so schnell, je nach der eingegebenen Frage (wo ich bei Basic 12,28 Sekunden auf eine bestimmte Antwort warten mußte, sind es jetzt nur noch 2,4 Sekunden). Das Programm könnte noch etwas schneller und kürzer sein, wenn ich es direkt in Comal geschrieben und nicht »wörtlich« aus Basic übersetzt hätte.

Menschenfreundliches und computerfreundliches Programmieren schließen einander also nicht prinzipiell aus. Comal beweist es. Den Namen Comal sollte man sich merken.

Wir waren einmal von der Frage ausgegangen, ob man auch in Basic strukturiert programmieren könne. Die Antwort war: Natürlich kann man.

Aber soll man auch? Die Antwort muß wiederum lauten: Natürlich soll man! Was für eine vernünftige Alternative gäbe es denn? Gut, es treten Probleme auf, vor allem, wenn ein Programm eine gewisse Länge überschreitet. Aber diese lassen sich lösen. Vorschläge dazu wurden gemacht. Die Alternative zum strukturierten Programmieren in Basic ist jedenfalls nicht unstrukturiertes Programmieren.

(Prof. Burkhard Leuschner/nj)

Hinweise zu »Comalchen«

Der Zweck des Programms COMALCHEN ist es, Fragen über Comal zu beantworten. Fragen können die Form von W-Fragen haben (Was ist..., Wieviel kostet...,Wer hat...) oder von Ja-Nein-Fragen (Ist..., kann..., bekommt man...).

Beobachtungen bei der Benutzung der ursprünglichen Version von COMALCHEN hatten gezeigt, daß die Benutzer selten beim Thema bleiben, entweder aus Vergeßlichkeit oder einfach aus Mutwillen.

Es ist also notwendig, nicht nur Eingaben zu berücksichtigen, die sich tatsächlich auf Comal beziehen, sondern (vor allem!) Eingaben, die sich nicht auf Comal beziehen. Daraus ergeben sich vier Typen von Eingaben:

1. W-Fragen, 2. Ja-Nein-Fragen, 3. Eingaben mit »du«,
4. Sonstige Eingaben: Aussagen (Nicht-Fragen)

Die Unterscheidung dieser Typen führt das Programm mit sehr einfachen Mitteln durch. Eingaben mit Fragezeichen am Ende sind Fragen. Wenn ein Fragewort enthalten ist, dann sind es W-Fragen, anderenfalls Ja-Nein-Fragen. Alles andere sind zunächst Aussagen. Wenn »du« enthalten ist, dann handelt es sich um eine Eingabe mit »du«, also um den dritten Typ. Alles andere gilt als Aussage des Typs 4. Die Antworttypen bestimmen sich nach mehreren Gesichtspunkten:

Typ 1. nach dem Fragetyp
Typ 2. nach der erfragten Information
Typ 3. danach, ob der Computer mit der Frage etwas anfangen konnte, ob er sie »verstand«
Typ 4. nach der Fragedisziplin des Benutzers

Die Antwortmöglichkeiten sind vom Programm vorgegeben. Sie können vom Computer nicht »erfunden«, sondern nur ausgewählt werden. Wo mir mehrere Antwortmöglichkeiten auf eine Frage einfielen, kann der Computer aus mehreren Antworten auswählen. Er tut dies nach dem Zufallsverfahren, achtet aber darauf, daß er erst alle Möglichkeiten erschöpft hat, ehe er eine schon einmal gegebene Antwort wiederholt. Die Antworten werden so gegeben, wie sie vorgegeben sind, mit einer Ausnahme: Manche Antworten sind »Rahmen«; der Rahmen ist vorgegeben, die Füllung des Rahmens hängt von der jeweiligen Eingabe ab, zum Beispiel »Was meinst du mit...?«

Die Auswahl einer Antwort geht so vor sich:

Auswahl nach dem Fragewort (Typ 1)

Jedem Fragetyp entspricht zunächst ein Antworttyp, das heißt W-Fragen werden mit Fakten beantwortet, Ja-Nein-Fragen mit »Ja«, »Nein«, »Zweifellos« oder ähnliche Eingaben, die »du« enthalten werden abgewehrt, sonstige Aussagen je nach Aussagetyp behandelt, »Adieu« führt zum Beispiel zur Beendigung des Programms.

Auswahl nach der erfragten Information, also nach dem Inhalt der Frage (Typ 2)

Wenn der Computer eine Frage überprüft, versucht er herauszufinden, worüber der Fragende etwas wissen will. Dazu sucht er nach »Schlüsselwörtern« in der Eingabe. Wenn zum Beispiel das Wort »Preis« enthalten ist, dann ist die Wahrscheinlickeit groß, daß der Benutzer wissen will, was COMAL kostet.

Wenn der Computer also eines dieser Schlüsselwörter findet, dann nimmt er an, daß die Frage den Preis betrifft. Er speichert dann »Preis« als Oberbegriff.

Es gibt im Datenmodul, getrennt durch Fragetypen, Gruppen solcher Oberbegriffe mit zugehörigen Schlüsselwörtern.

Wenn der Computer nun eine passende Antwort sucht, dann begibt er sich zunächst zu dem entsprechenden Antworttyp, zum Beispiel den Faktenantworten. Dort sucht er nach dem gespeicherten Oberbegriff, also zum Beispiel »Preis«. Aus den zugehörigen Antwortmöglichkeiten wählt er dann eine aus.

Der Computer versteht die Frage nicht (Typ 3)

Wenn der Computer keinen Oberbegriff gefunden hatte, kann er mit der Eingabe nichts anfangen. Wenn er eine Frage nicht verstanden hat, antwortet er mit »Ich weiß nicht« (oder einer alternativen Möglichkeit); wenn er eine Aussage nicht verstanden hat, sagt er: »Ich habe es nicht verstanden«.

Der Benutzer verirrt sich (Typ 4)

Manche Benutzer vergessen völlig, was das Thema ist. Um den Fragenden wieder auf die richtige Fährte zu bringen, wird gezählt, wie oft eine Eingabe nicht verstanden wurde. Nach dem dritten Mal wird an das Thema erinnert. Wenn der Benutzer trotzdem weiter das Thema verfehlt, bricht der Computer nach dem sechsten Mal die Unterhaltung ab.

Wie »natürlich« die Unterhaltung wirkt, hängt von der Logik des Fragers ab, aber auch von den Daten, das heißt den Schlüsselwörtern und den zugeordneten Antwortmöglichkeiten. Da kann noch unendlich viel verbessert und ergänzt werden. Normal ist, daß einem bei jedem Programmlauf eine neue Idee einfällt. Versuchen Sie zum Beispiel, eine Hilfe für denjenigen Benutzer einzubauen, der nicht mehr weiß, worüber er Fragen stellen könnte.

Falls Sie eigene Daten eingeben wollen — die Zahlen hinter den Oberbegriffen der Antwortdaten dienen dazu, die Liste zu führen, die registriert, welche Antwortmöglichkeiten jeweils schon »verbraucht« sind. Diese Zahlen brauchen nicht in numerischer Reihenfolge zu sein. Wenn Sie irgendwo eine neue Frage plus Antwortgruppe einfügen, geben Sie ihr einfach die nächste laufende Nummer. Hauptsache, jede Antwortgruppe hat ihre eigene Zahl. Achten Sie bei der Dimensionierung des Feldes LISTE$( ) darauf, daß das Feld auch groß genug ist.

Natürlich können Sie alle meine Daten hinauswerfen und eigene eingeben. Achten Sie aber bitte darauf, daß die Grundgruppierungen mit den bisherigen Zeilennummern erhalten bleiben, denn darauf springt das Programm mit RESTORE zu.

Tabelle 2. Anmerkungen zum Programm »COMALCHEN«

```
10 REM PROGRAMM: COMALCHEN
20 REM VERSION:  1.1
25 REM DATUM:    27.2.86
30 REM COMPUTER: C128
40 REM SPRACHE:  BASIC 7.0
50 REM AUTOR:    BURKHARD LEUSCHNER
60 :
100 REM *******************************
110 REM ZENTRALMODUL
130 : GOSUB 1000 VORBEREITUNG
140 : GOSUB 2000 BEGRUESSUNG
150 :
160 : DO
170 :  GOSUB 3000 FRAGE
180 :  GOSUB 4000 ANTWORT
190 : LOOP UNTIL ADIEU
200 :
210 : GOSUB 5000 ABSCHIED
220 : GOSUB 6000 PROGRAMMENDE
240 REM ENDE ZENTRALMODUL
250 REM *******************************
260 :
270 :
1000 REM MODUL VORBEREITUNG
1010 : GOSUB 50000 SYSTEM KONFIGURATION
1020 : GOSUB 51000 INITIALISIERUNGEN
1030 : GOSUB 46000 FUNKTIONEN
1040 : GOSUB 52000 DATEN LESEN
1050 RETURN
1060 REM ENDE MODUL VORBEREITUNG
1070 :
2000 REM MODUL BEGRUESSUNG
2010 : GOSUB 10000 VORSTELLUNG
2020 : GOSUB 10200 BEGRUESSEN
2030 : GOSUB 10500 BEGINN UNTERHALTUNG
2040 RETURN
2050 REM ENDE MODUL BEGRUESSUNG
2060 :
3000 REM MODUL FRAGE
3010 : GOSUB 11000 EINGABE FRAGE
3015 REM FAST
3020 : GOSUB 11300 FRAGENTYP FESTSTELLEN
3030 : GOSUB 11500 SCHLUESSELWORT SUCHEN
3040 RETURN
3050 ENDE MODUL FRAGE
3060 :
4000 REM MODUL ANTWORT
4010 : GOSUB 15000 ANWORTTYP FESTLEGEN
4020 : GOSUB 15400 ANTWORT WAEHLEN
4025 REM SLOW
4030 : GOSUB 15600 ANTWORT AUSGEBEN
4040 RETURN
4050 REM ENDE MODUL ANTWORT
4060 :
4070 :
5000 REM MODUL ABSCHIED
5020 : GOSUB 29000 ABSCHIEDSKOMMENTAR
5030 RETURN
5040 REM ENDE MODUL ABSCHIED
5050 :
6000 REM MODUL PROGRAMMENDE
6005 : PRINTCHR$(12)
6010 : END
6020 ENDE MODUL PROGRAMMENDE
6030 :
6040 REM *******************************
6050 REM *******************************
6060 :
10000 REM NEBENMODUL BEGRUESSUNG
10010 REM ------------------------
10020 REM U-MODUL VORSTELLUNG
10025 REM EXPORT: NAME$
10030 : PRINT CHR$(147)
10040 : PRINT "ICH BIN COMALCHEN - UND WER BIST DU?
10050 : PRINT
10060 : DO
10070 :  PRINT
10080 :  INPUT NAME$
10090 :  PRINT "IST ";NAME$;" DEIN VORNAME?"
10100 :  INPUT MJN$
10110 :  IF INSTR("NN",LEFT$(MJN$,1)) THEN BEGIN
10120 :   PRINT
10130 :   PRINT"ICH MOECHT' ABER DOCH GERN DEINEN(7SPACE)VORNAMEN WISSEN!"
10133 :   PRINT
10135 :   PRINT"WIE HEISST DU ALSO?"
10140 :  BEND
10150 : LOOP UNTIL INSTR("JJ",LEFT$(MJN$,1))
10160 RETURN
10170 REM ENDE U-MODUL VORSTELLUNG
10180 :
10200 REM U-MODUL BEGRUESSEN
10210 REM IMPORT: NAME$
10220 : DO
10230 :  PRINT
10240 :  PRINT"GRUESS GOTT, ";NAME$;"."
10250 :  PRINT
10260 :  INPUT MEINGABE$
10270 :  UEINGABE$=MEINGABE$
10280 :   GOSUB 40200 LOWERCASE
10290 :   GOSUB 40400 SONDERZEICHEN
10300 :  MEINGABE$=UEINGABE$
10310 :  RESTORE 55830:REM GRUSS
10320 :  DO
10330 :   READ MGRUSS$
10340 :   IF NOT(MGRUSS$="%") THEN BEGIN
10350 :    IF INSTR(MEINGABE$,MGRUSS$) THEN MOK=TRUE
10355 :   BEND
10360 :  LOOP UNTIL MOK OR MGRUSS$="%"
10370 : LOOP UNTIL MOK
10380 RETURN
10390 REM ENDE U-MODUL BEGRUESSEN
10400 :
10500 REM U-MODUL BEGINN UNTERHALTUNG
10510 : PRINT
10520 : PRINT"DU WILLST DICH ALSO MIT MIR UEBER COMAL UNTERHALTEN.
10530 : PRINT
10540 : PRINT"OK. SCHIESS LOS. WAS WILLST DU WISSEN?
10550 : PRINT
10670 RETURN
10680 ENDE U-MODUL BEGINN UNTERHALTUNG
10700 REM ENDE NEBENMODUL BEGRUESSUNG
10710 :
10720 :
11000 REM NEBENMODUL FRAGE
11010 REM ------------------
11020 REM U-MODUL EINGABE FRAGE
11030 REM EXPORT: FRAGE$,EINGABE$
11040 : GOSUB 12000 EINGABE
11050 : EINGABE$=FRAGE$
11060 : UEINGABE$=EINGABE$
11070 :  GOSUB 12500 FORM STANDARDISIEREN
11080 : EINGABE$=UEINGABE$
11090 RETURN
11100 REM ENDE U-MODUL EINGABE FRAGE
11110 :
11300 REM U-MODUL FRAGENTYP FESTSTELLEN
11310 REM IMPORT: EINGABE$
11320 REM EXPORT: TPEINGABE
11330 : TPEINGABE=0
11340 : IF RIGHT$(EI$,1)="?" THEN BEGIN
11350 :  GOSUB 13000 FRAGEPRONOMEN
11360 :  IF MGEFUNDEN THEN TPEINGABE=1 : ELSE TPEINGABE=2
11370 :  BEND: ELSE TPEINGABE=4
11380 : IF TP=4 THEN IF FN DU DRIN(0) THEN TP=3
11390 RETURN
11400 REM ENDE U-MODUL FRAGETYP FESTSTELLEN
11410 :
11500 REM U-MODUL SCHLUESSELWORT SUCHEN
11510 REM IMPORT: TPEINGABE
11520 REM EXPORT: OBERBEGRIFF$,KY
11530 : GOSUB 14000 SCHLUESSELGRUPPE SUCHEN
11540 : GOSUB 14500 OBERBEGRIFF FINDEN
11550 RETURN
11560 REM ENDE U-MODUL SCHLUESSELWORT SUCHEN
11570 :
12000 REM PROC: EINGABE
12010 REM EXPORT: FRAGE$
12020 : DO
12025 :  FRAGE$=""
12027 :   REM POKE 208,3:POKE 7,34:POKE 7+1,34:POKE 7+2,20: REM TASTATURPUFFER
12028 :   REM SOLL ANFUEHRUNGSZEICHEN            BEIM INPUT ERZEUGEN
12030 :  INPUT FRAGE$
12040 :  IF FRAGE$="" THEN PRINT"NAST DU WAS GESAGT?"
12050 : LOOP UNTIL FRAGE$>""
12060 RETURN
12070 :
12500 REM PROC: FORM STANDARDISIEREN (UEINGABE$:IN/OUT)
12510 : GOSUB 40000 LEERZEICHEN
12520 : GOSUB 40200 LOWERCASE
12530 : GOSUB 40400 SONDERZEICHEN
12540 RETURN
12550 :
13000 REM PROC: FRAGEPRONOMEN
13010 REM IMPORT: EINGABE$, PRONOMEN$(),ZAHLPRONOMEN
13020 REM EXPORT: MGEFUNDEN,WH
13030 : MGEFUNDEN=FALSE:WH=FALSE
13040 : FOR UI=1 TO ZAHL
13050 :  IF RIGHT$(EINGABE$,LEN(PR$(UI))+1)=PR$(UI)+"?" THEN WH=TRUE
13060 :  IF INSTR(EI$,PR$(UI)) THEN MG=TRUE: UI=ZAHL
13070 : NEXT
13080 RETURN
13090 :
14000 REM PROC SCHLUESSELGRUPPE SUCHEN
14010 REM IMPORT: TPEINGABE
14020 : REM CASE
14030 : ON TPEINGABE GOTO 14040,14060,14080,14100:GOTO 14120
14040 :  RESTORE 55000:REM W-FRAGEN
14050 : GOTO 14120
14060 :  RESTORE 55400:REM J/N-FRAGEN
14070 : GOTO 14120
14080 :  RESTORE 55600:REM DU
14090 : GOTO 14120
14100 :  RESTORE 55800:REM AUSSAGE
14110 : GOTO 14120
14120 : REM ENDCASE
14130 RETURN
14140 :
14500 REM PROC: OBERBEGRIFF FINDEN
14510 REM IMPORT: EINGABE$
14520 REM EXPORT: OBERBEGRIFF$,KY
14530 : DO
14540 :  READ OBERBEGRI FF$
14550 :  IF NOT(OB$="@@@")THEN BEGIN
14560 :   DO
14570 :    READ USCHLUESSL$
14580 :    IF NOT(US$="%") THEN BEGIN
14590 :     UP OS=INSTR(US$,"%")
14600 :     IF UP=0 THEN KY=(INSTR(EINGABE$,US$)<>0):ELSE BEGIN
14610 :      UWO RT$(1)=LEFT$(US$,UP-1)
14620 :      UWO RT$(2)=MID$(US$,UP+1)
14630 :      KY=(INSTR(EI$,UW$(1)) AND INSTR(EI$,UW$(2))<>0)
14640 :     BEND
14650 :    BEND
14660 :   LOOP UNTIL KY OR US$="%"
14670 :  BEND
14680 : LOOP UNTIL KY OR OB$="@@@"
14690 RETURN
14700 REM ENDE NEBENMODUL FRAGE
14710 :
14720 :
15000 REM NEBENMODUL ANTWORT
15010 REM -------------------
15020 REM U-MODUL ANTWORTTYP FESTLEGEN
15030 REM IMPORT: TPEINGABE,KY
15040 REM EXPORT: TYPANTWORT,OBERBEGRIFF$
15050 : REM CASE
15060 : ON TPEINGABE GOTO 15070,15110,15130,15150:GOTO15160
15070 :  IF WH THEN KY=0:OB$="NUR FRAGEPRONOMEN":TYPANTW=1:NI=0:ELSE BEGIN
15080 :   IF KY THEN TYP=1:NICHTTHEMA=0:ELSE TYP=5:NI=NI+1:OB$="WEISSNICHT"
15090 :  BEND
15100 : GOTO 15160
15110 :  IF KY THEN TYP=2:NI=0:ELSE TYP=5:NI=NI+1:OBERBEGRI FF$="WEISSNICHT
15120 : GOTO 15160
15130 :  TYP=3:NI=NI+1
15140 : GOTO 15160
15150 :  IF KY THEN TYP=4:NI=0:ELSE TYP=6:NI=NI+1:OB$="VERSTEHNICHT"
15160 : REM ENDCASE
15170 :
15180 : IF NI>6 THEN TYPANTW=8:OBERBEGR$="ABBRUCH":ADIEU=TRUE:ELSE BEGIN
15190 :  IF NICHTTHEMA>3 THEN TYPANTW=7:OBERBEGRI FF$="THEMAVERFEHLT"
15200 : BEND
15210 : IF OB$="ABSCHIED" THEN ADIEU=TRUE
15220 RETURN
15230 REM ENDE U-MODUL ANTWORTTYP FESTLEGEN
15240 :
15400 REM U-MODUL ANTWORT WAEHLEN
15410 REM IMPORT: TYPANTWORT
15415 REM EXPORT: ANTWORT
15420 : GOSUB 16000 ANTWORTGRUPPE FINDEN
15430 : GOSUB 43000 ANTWORT AUSSUCHEN
15440 RETURN
15450 REM ENDE U-MODUL ANTWORT WAEHLEN
15460 :
15600 REM U-MODUL ANTWORT AUSGEBEN
15610 REM IMPORT: ANTWORT$
15620 : GOSUB 44000 ANWORT ERGAENZEN
15630 : GOSUB 45000 ANTWORT AUF BILDSCHIRM
15640 RETURN
15650 REM ENDE U-MODUL ANTWORT AUSGEBEN
15660 :
16000 REM PROC ANTWORTGRUPPE FINDEN
16010 REM IMPORT: TYPANTWORT
16020 : REM CASE
16030 : ON TYPANTW GOTO 16050,16070,16090,16110,16130,16150,16170,16190
16040 : GOTO16200
16050 :  RESTORE 56000:REM FAKTENANTW
16060 : GOTO 16200
16070 :  RESTORE 56500:REM J/N-ANTW
16080 : GOTO 16200
16090 :  RESTORE 56700:REM DU
16100 : GOTO 16200
16110 :  RESTORE 56800:REM AUSSAGE
16120 : GOTO 16200
16130 :  RESTORE 57000:REM WEISS NICHT
16140 : GOTO 16200
16150 :  RESTORE 57300:REM VERSTEH NICHT
16160 : GOTO 16200
16170 :  RESTORE 57500:REM THEMA VERFEHLT
16180 : GOTO 16200
16190 :  RESTORE 57800:REM ABBRUCH
16200 : REM ENDCASE
16220 RETURN
16240 REM ENDE NEBENMODUL ANTWORT
16250 :
16260 :
29000 REM NEBENMODUL ABSCHIED
29010 REM --------------------
29020 REM U-MODUL ABSCHIEDSKOMMENTAR
29030 : RESTORE 57900
29040 : OBERBEGRI FF$="ENDE"
29050 : GOSUB 43000 ANTWORT AUSSUCHEN
29060 : GOSUB 44000 ANTWORT ERGAENZEN
29070 : GOSUB 45000 ANTWORT AUF BILDSCHIRM
29080 RETURN
29090 REM ENDE U-MODUL ABSCHIEDSKOMMENTAR
29110 ENDE NEBENMODUL ABSCHIED
29120 :
29130 :
40000 REM PROZEDURENMODUL
40010 REM ================
40020 REM PROC LEERZEICHEN (UEINGABE$: IN/OUT)
40030 : DO WHILE LEFT$(UEINGABE$,1)=" ":UE$=MID$(UE$,2):LOOP
40040 : DO WHILE RIGHT$(UE$,1)=" ":UE$=LEFT$(UE$,LEN(UE$)-1):LOOP
40050 : DO WHILE INSTR(UE$,"(2SPACE)")
40060 :  UP=INSTR(UE$,"(2SPACE)")
40070 :  UE$=LEFT$(UE$,UP-1)+MID$(UE$,UP+1)
40080 : LOOP
40090 RETURN
40100 :
40200 REM PROC: LOWERCASE (UEINGABE$: IN/OUT)
40205 REM IMPORT KLEIN$,GROSS$
40210 : FOR UI=1 TO LEN(UE$)
40220 :  IF INSTR(GR$,MID$(UE$,UI,1)) THEN BEGIN
40230 :   UP=INSTR(GR$,MID$(UE$,UI,1))
40240 :   UE$=LEFT$(UE$,UI-1)+MID$(KL$,UP,1)+MID$(UE$,UI+1)
40250 :  BEND
40260 : NEXT
40270 RETURN
40280 :
40400 REM PROC: SONDERZEICHEN (UEINGABE$: IN/OUT)
40410 : UUMLAUTE$     ="\[]{|}~"
40420 : USUBSTITUTES$="SSAEOEUE"
40430 : FOR UI=1 TO LEN(UUM$)
40440 :   UZEICHEN$=MID$(UUM$,UI,1)
40450 :  DO WHILE INSTR(UE$,UZ$)
40460 :   UP=INSTR(UE$,UZ$)
40470 :   UE$=LEFT$(UE$,UP-1)+MID$(USUB$,UI*2-1,2)+MID$(UE$,UP+1)
40480 :  LOOP
40490 : NEXT
40500 RETURN
40510 :
43000 REM PROC ANTWORT AUSSUCHEN
43010 REM IMPORT: OBERBEGRIFF$,LISTE$(),NULL$
43020 REM EXPORT: ANTWORT$,LISTE$()
43030 : UZAEHLER=0
43040 : DO
43050 :  READ UTITEL$
43060 :  IF NOT(UT$="@@@") THEN BEGIN
```

Listing 1. Comalchen — das vollständige Programm

```
43070 :    IF UT$=OB$ THEN BEGIN
43080 :     READ UNR LIS TE
43090 :     DO
43100 :      UZ=UZ+1
43110 :      READ UANTW$(UZ)
43120 :     LOOP UNTIL UANTW$(UZ)="&"
43130 :     UZ=UZ-1
43140 :    BEND
43150 :   BEND
43160 : LOOP UNTIL UANTW$(UZ+1)="&" OR UT$="eee"
43170 :
43180 : IF INSTR(LIS TE$(UNR),"0")=0 THEN LI$(UNR)=LEFT$(NULL$,UZ)
43190 :
43200 : DO
43210 :  UP=FN ZUFALL(UZ)
43220 : LOOP UNTIL MID$(LI$(UNR),UP,1)="0"
43230 :
43240 : ANTWO RT$=UANTW$(UP)
43250 : LI$(UNR)=LEFT$(LI$(UNR),UP-1)+"1"+MID$(LI$(UNR),UP+1)
43260 RETURN
43270 :
44000 REM PROC ANTWORT ERGAENZEN
44010 REM IMPORT: ANTWORT$,FRAGE$,NAME$
44020 REM EXPORT: ANTWORT$
44030 : IF INSTR(ANTW$,"+") THEN BEGIN
44040 :  UP=INSTR(AN$,"+NAME")
44050 :  IF UP THEN AN$=LEFT$(AN$,UP-1)+NAME$+MID$(AN$,UP+5)
44060 :  UP=INSTR(AN$,"+FRAGE")
44070 :  IF UP THEN AN$=LEFT$(AN$,UP-1)+FRAGE$+MID$(AN$,UP+6)
44080 : BEND
44090 RETURN
44100 :
45000 REM PROC: ANTWORT AUF BILDSCHIRM
45010 REM IMPORT: ANTWORT$
45020 : PRINT " ";ANTWO RT$
45030 : PRINT
45040 RETURN
45060 REM ENDE PROZEDURENMODUL
45070 :
45080 :
46000 REM FUNKTIONENMODUL
46010 REM ================
46020 REM FUNC DU DRIN
46025 : REM IMPORT: EINGABE$
46030 : DEF FN DU(X)=ABS(INSTR(EI$,"DU ")>0 OR INSTR(EI$," DU")>0)
46040 REM ENDFUNC
46050 :
46060 REM FUNC ZUFALLSZAHL(OBERGRENZE:IN)
46070 : UDUMMY=RND(-TI)
46080 : DEF FN ZUFALL(OB)=INT(RND(1)*OB)+1
46090 REM ENDFUNC
46100 :
46110 RETURN
46120 REM ENDE FUNKTIONENMODUL
46130 :
46140 :
50000 REM NEBENMODUL VORBEREITUNG
50010 REM ------------------------
50020 REM U-MODUL SYSTEM KONFIGURATION
50030 : PRINT CHR$(14):PRINTCHR$(11)
50040 : COLOR 0,9:COLOR 4,9:COLOR 5,2
50090 RETURN
50100 REM ENDE U-MODUL SYSTEM KONFIGURATION
50110 :
51000 REM U-MODUL INITIALISIERUNGEN
51010 : NULL$="0000000000000000000000"
51020 : GROSS$="ABCDEFGHIJKLMNOPQRSTUVWXYZ+-="
51030 : KLEIN$="ABCDEFGHIJKLMNOPQRSTUVWXYYPCX"
51100 : TRUE=1:FALSE=0
51110 : ADIEU=FALSE
51120 : ZAHLPRO NOMEN=13
51200 : DIM PRO NOMEN$(ZAHL),LIS TE$(40), UANTWO RT$(20)
51970 RETURN
51980 REM ENDE U-MODUL INITIALISIERUNGEN
51990 :
52000 REM U-MODUL DATEN LESEN
52010 : RESTORE 58900: REM FRAGEPRONOMEN
52020 : FOR UI=1 TO ZAHL: READ PRO NOMEN$(UI):NEXT
52030 RETURN
52040 REM ENDE U-MODUL DATEN LESEN
52050 REM ENDE NEBENMODUL VORBEREITUNG
52060 :
53990 :
55000 REM DATENMODUL
55005 REM ::::::::::
55010 REM SCHLUESSELWOERTER
55020 REM ------------------
55030 REM 1: W-FRAGEN
55040 DATA THEMA
55045 DATA THEMA,WORUEBER,&
55050 DATA PROGRAMMIERSPRACHE
55060 DATA WAS&PROGRAMMIERSPRACHE,&
55080 DATA ALTER
55090 DATA WIE ALT,WANN,&
55100 DATA ERFINDER
55110 DATA WERERFUNDEN,WEM&ERFUNDEN,WEM&ENTWICKELT,WER&ENTWICKELT,ERFIND,&
55120 DATA QUALITAET
55130 DATA GUT,EIGENSCHAFT,BESONDER,VORTEIL,SCHNELL,&
55140 DATA HERKUNFT
55150 DATA WOHER,WO&HER,LAND,&
55160 DATA LITERATUR
55170 DATA LITERATUR,INFO,LESEN,ERFAHR,BUCH,BUECH,&
55180 DATA PREIS
55190 DATA KOST,PREIS,WIEVIEL,TEUER,&
55200 DATA QUELLE
55210 DATA BEZIEH,BEZUGS,BEZOG,BEKOMM,KRIEG,KAUF,VON WEM,&
55220 DATA COMALGRUPPEN
55230 DATA GRUPPE,ADRESS,USER,GROUP,&
55240 DATA IMPLEMENTIERUNG
55250 DATA COMPUTER,MASCHIN,SYSTEM,&
55260 DATA VERSIONEN
55270 DATA VERSION,&
55280 DATA NAME
55290 DATA HEISS&COMAL,BEDEUTET&COMAL,&
55300 DATA STRUKTURIERT
55310 DATA STRUKTURIERT,PROGRAMMIEREN,PROGRAMME,BEFEHL,&
55320 DATA WARUM NICHT
55330 DATA WARUM&NICHT,&
55340 DATA NICHTVERSTANDEN
55350 DATA WIE&BITTE,WAS&MEINST,WAS&SAG,,SOLL&HEISS
55351 DATA &
55355 DATA DEFINITION
55356 DATA WAS&COMAL,&
55360 DATA eee
55370 :
55400 REM 2: JA/NEIN-FRAGEN
55410 DATA QUALITAET
55420 DATA GUT,BESSER&BASIC,BESSER&PASCAL,VORTEIL,&
55430 DATA LITERATUR
55440 DATA LITERATUR,INFO,LESEN,ERFAHR,BUCH,BUECH,ARTIKEL,&
55450 DATA PREIS
55460 DATA PREIS,VIEL,TEUER,&
55470 DATA KAUF
55480 DATA KAUF,SCHENK,&
55490 DATA VERSIONEN
55500 DATA DISK,KASSET,CASS,CC,VERSION,MODUL,ROM,"RAM "," RAM",&
55510 DATA STRUKTURIERT
55520 DATA STRUKTUR,&
55530 REM NACHFRAGE
55540 DATA MEINSTDU
55550 DATA MEINST&DU,GLAUBST&DU,SICHER,WIRKLICH,&
55555 DATA COMALGRUPPEN
55556 DATA GRUPPE,ADRESS,USER,GROUP,&
55560 DATA eee
55570 :
55600 REM 3: DU
55610 DATA BOESE
55620 DATA WAS&WEISST,DU&WEISST,',&
55630 DATA DU
55640 DATA DU,&
55650 DATA eee
55660 :
55800 REM 4: AUSSAGE
55810 DATA ABSCHIED
55820 DATA ADE,ADIEU,WIEDERSEH,ENDE,AUFHOER,BIS&SPAET,TSCHUES,TSCHAU,&
55830 DATA GRUSS
55840 DATA TAG, MORGEN,ABEND,HALLO,GRUESS,GOTT,GRUSS,&
55850 DATA AUSRUF
55860 DATA ',&
55870 DATA JA
55880 DATA JA,MHM,ICH&WEISS,&
55890 DATA NEIN
55900 DATA NEIN,NICHT&VERSTEH,NICHT&VERSTAND,&
55910 DATA DANK
55920 DATA DANK,THANK,MERCI,&
55970 DATA eee
55980 :
56000 REM ANTWORTEN
56005 REM ---------
56010 REM 1: FAKTENANTWORTEN
56015 DATA DEFINITION,1
56020 DATA "COMAL IST EINE PROGRAMMIERSPRACHE."
56025 DATA &
56030 DATA PROGRAMMIERSPRACHE,2
56035 DATA"EINE PROGRAMMIERSPRACHE DIENT DAZU, EI-NEN COMPUTER ZU PROGRAMMIERE
N.
56040 DATA &
56045 DATA ALTER,3
56050 DATA "COMAL IST KEINESWEGS NEU."
56055 DATA "COMAL IST SCHON 13 JAHRE ALT."
56060 DATA "COMAL WURDE SCHON 1973 ENTWICKELT."
56065 DATA &
56070 DATA ERFINDER,4
56075 DATA "DER 'ERFINDER' VON COMAL IST BORGE(6SPACE)CHRISTENSEN."
56080 DATA "COMAL WURDE VON B. CHRISTENSEN UND(6SPACE)B. IDEFSTEDT ENTWICKELT.
"
56085 DATA &
56090 DATA QUALITAET,5
56095 DATA "COMAL IST EINDEUTIG DIE PROGRAMMIER-(4SPACE)SPRACHE DER ZUKUNFT."
56100 DATA "COMAL IST SO GUT WIE BASIC UND PASCAL(2SPACE)ZUSAMMEN."
56105 DATA "COMAL HAT DIE GUTEN EIGENSCHAFTEN VON(3SPACE)BASIC UND PASCAL."
56110 DATA "WENN DU COMAL HAST, DANN KANNST DU(5SPACE)BASIC VERGESSEN."
56115 DATA "ANDERE SPRACHEN WIE FORTH, ELAN, PASCALVERBLASSEN VOR COMAL."
56120 DATA "COMAL IST EINE SPRACHE, DIE V.A. AUCH(2SPACE)DEN MENSCHEN ERNST NI
MMT."
56125 DATA "COMAL IST Z.B. MENSCHENFREUNDLICH."
56130 DATA &
56135 DATA HERKUNFT,6
56140 DATA "COMAL KOMMT AUS DAENEMARK."
56145 DATA "DIE ERFINDER VON COMAL SIND DAENEN."
56150 DATA "COMAL KOMMT, WIE AUCH ANDERE GUTE SOFT- WARE, AUS EUROPA."
56155 DATA &
56160 DATA LITERATUR,7
56165 DATA "ES GIBT EINE GANZE MENGE LITERATUR,(4SPACE)V.A. AUF ENGLISCH."
56170 DATA "DER ERFINDER VON COMAL HAT MEHRERE(6SPACE)BUECHER GESCHRIEBEN."
56175 DATA "MEIKO MOLGAST UEBERSETZTE(2SPACE)Z.B.(8SPACE)CHRISTENSENS 'C. FROM
A-Z'"
56180 DATA "DAS COMAL 0.14 HANDBUCH, Z.B.
56185 DATA"FUER ANFAENGER: CHRISTENSEN, 'STRUKTU- RIERTE PROGRAMMIERUNG MIT C.
''"
56190 DATA "ALLES UEBER COMAL ENTHAELT DAS 'COMAL(2SPACE)HANDBOOK' VON LEN LIN
DSAY."
56195 DATA "VERA BIRKENBIHL SCHRIEB 1984 DAS BUCH 'VON BASIC ZU COMAL'."
56200 DATA "1986 ERSCHEINT VON VOLKER FISCHER DAS(2SPACE)BUCH 'COMAL IN BEISPI
ELEN'."
56205 DATA "IM '64ER' FINDET MAN SEIT AUGUST 1984(2SPACE)GELEGENTLICH INFORMAT
IONEN.
56210 DATA &
56215 DATA PREIS,8
56220 DATA"COMAL 0.14 KOSTET FAST NICHTS, DAS(6SPACE)STECKMODUL (V. 2.0) DM 19
8."
56225 DATA "DAS STECKMODUL MIT COMAL 2.01 KOSTET DM 198.-, 0.14 EIN PAAR MARK.
"
56230 DATA &
56235 DATA QUELLE,9
56240 DATA "BEZUGSQUELLEN SIND V.A. DIE BEIDEN(5SPACE)DEUTSCHEN COMAL GRUPPEN.
56245 DATA &
56250 DATA COMALGRUPPEN,10
56255 DATA "DHR. SANISIUS, FREIHEITSTR. 30, 4000 B'D 12 / B. BELZ, 2270 UTERSU
M
56260 DATA &
56265 DATA IMPLEMENTIERUNG,11
56270 DATA "COMAL GIBT'S V.A. FUER COMMODORE COMPUTER."
56275 DATA "COMAL GIBT'S AUCH FUER DEN IBM PC UND(2SPACE)KOMPATIBLE."
56280 DATA "AB FRUEHJAHR GIBT'S COMAL FUER 2.0 FAST ALLE GAENGIGEN COMPUTER.
56285 DATA &
56290 DATA VERSIONEN,12
56295 DATA "DIE NEUESTE VERSION IST 2.0 (Z.B.ALS(3SPACE)STECKMODUL FUER DEN C6
4)."
56300 DATA "DIE AELTESTE VERSION IST COMAL 0.11(4SPACE)(1981 FUER DEN PET)."
56305 DATA "DIE PREISWERTESTE VERSION IST IM AUGEN- BLICK DIE 0.14 (C64)."
56310 DATA &
56315 DATA NAME,13
56320 DATA "'COMAL' BEDEUTET 'COMMON BLGORITHMIC LANGUAGE'."
56325 DATA "WAS DER NAME BEDEUTET, DARAUF KOMMT'S(2SPACE)EIGENTLICH GAR NICHT
AN."
56330 DATA "WICHTIG IST, WAS COMAL TUT, NICHT WAS(2SPACE)DER NAME MEINT."
56335 DATA "'COMAL' IST EINE ABKUERZUNG. ABER DIE(2SPACE)MACHT NICHT VIEL SINN
."
56340 DATA &
56345 DATA STRUKTURIERT,14
56350 DATA "SO PROGRAMMIEREN, DASS AUCH EIN MENSCH DAS PROGRAMM VERSTEHT."
56355 DATA "UEBERSICHTLICH PROGRAMMIEREN."
56360 DATA "BEIM PROGRAMMIEREN AUCH AN DEN MENSCHENDENKEN.
56365 DATA &
56370 DATA WARUM NICHT,15
56375 DATA "HEIL!","NUN JA ...","DAS WEISST DU SELBER SEHR WOHL."
56380 DATA &
56385 DATA NICHTVERSTANDEN,16
56390 DATA "WAR ICH SO UNKLAR?","DU HAST MICH NICHT VERSTANDEN?"
56395 DATA "HAB ICH MICH SO UNDEUTLICH AUS-(8SPACE)GEDRUECKT?"
56400 DATA "ALSO, DEUTLICHER KANN ICH'S NICHT MEHR SAGEN."
56405 DATA &
56410 DATA NUR FRAGEPRONOMEN,17
56415 DATA "+FRAGE(LEFT) WAS?","+FRAGE","WAS MEINST DU MIT '+FRAGE(LEFT)'?"
56420 DATA "SCHWIERIG, SCHWIERIG ..."
56425 DATA "ALS COMPUTER HABE ICH PROBLEME, SOLCHE(2SPACE)KURZFRAGEN ZU VERSTE
HEN."
56430 DATA &
56435 DATA THEMA,18
56440 DATA "COMAL NATUERLICH.","HIE BISHER: COMAL."
56445 DATA "IMMER NOCH COMAL.","GLAUB'S ODER GLAUB'S NICHT: COMAL!!!"
56450 DATA &
56485 DATA eee
56490 :
56500 REM 2: JA/NEIN-ANTWORTEN
56505 DATA STRUKTURIERT,19
56510 DATA "WER STRUKTURIERT PROGRAMMIEREN WILL,DERIST MIT COMAL GUT BEDIENT.
56515 DATA "WENN DU STRUKTRURIERT DENKST, HILFT(4SPACE)COMAL BEIM ADDIEREN."
56520 DATA "COMAL HAT PROZEDUREN UND FUNKTIONEN, UMNUR EINIGES ZU NENNEN."
56525 DATA "COMAL HAT REPEAT-, WHILE- UND FOR-(6SPACE)SCHLEIFEN, Z.B."
56530 DATA "COMAL STELLT U.A. IF- UND CASE-BLOECKE ZUR VERFUEGUNG."
56535 DATA "MIT COMAL STRUKTURIERT ZU PROGRAMMIEREN IST EIN GEDICHT."
56540 DATA &
56545 DATA QUALITAET,20
56550 DATA "JA.","EINDEUTIG.","OHNE ZWEIFEL."
56555 DATA &
56560 DATA LITERATUR,21
56565 DATA "JA, EINE GANZE MENGE, UND NICHT NUR AUFENGLISCH, AUCH AUF DEUTSCH.
56570 DATA &
56575 DATA PREIS,22
56580 DATA "KANN MAN EIGENTLICH NICHT SAGEN.","NICHT RECHT EIGENTLICH."
56585 DATA &
56590 DATA KAUF,23
56595 DATA "DIE VERSION 0.14 BEKOMMT MAN FAST GE-(2SPACE)SCHENKT, DIE ANDERN K
OSTEN.
56600 DATA &
56605 DATA VERSIONEN,24
56610 DATA "ES GIBT VERSIONEN AUF DISKETTE UND IM(2SPACE)ROM (Z.B. C64-STECKMO
DUL)."
56615 DATA &
56620 REM BESTAETIGUNG AUF NACHFRAGE
56625 DATA MEINSTDU,25
56630 DATA "ICH DENK SCHON.","WARUM NICHT?","WAS SOLLTE DAGEGEN SPRECHEN?"
56635 DATA "GEWISS DOCH.","ABER JA DOCH."
56640 DATA &
56642 DATA COMALGRUPPEN,26
56643 DATA "DHR. SANISIUS, FREIHEITSTR. 30, 4000 B'D 12 / B. BELZ, 2270 UTERSU
M
56644 DATA &
```

Listing 1. »Comalchen« (Fortsetzung)

```
56645 DATA ###
56650 :
56700 REM 3: 'DU'-ANTWORT
56705 DATA DU,27
56710 DATA "ICH BIN VOELLIG UNWICHTIG."
56715 DATA "MICH KANNST DU GERN VERGESSEN."
56720 DATA "ICH BIN DOCH NICHT DAS THEMA."
56725 DATA "LASS RUHIG MICH AUS DEM SPIEL."
56730 DATA "WIR WOLLTEN DOCH NICHT UEBER MICH(6SPACE)REDEN."
56735 DATA &
56740 :
56745 DATA BOESE,28
56750 DATA "ABER, ICH BITTE DICH!","REISS DICH ZUSAMMEN!"
56755 DATA "MEINST DU DOCH NICHT IM ERNST, ODER?"
56760 DATA "SO WAS SAGT MAN DOCH NICHT!!!"
56765 DATA "WAS FICHT DICH PLOETZLICH AN?"
56770 DATA &
56775 DATA ###
56780 :
56800 REM 4: AUSSAGEN
56810 DATA GRUSS,29
56820 DATA "ICH FUERCHTE, DU HAST VORHIN(11SPACE)GESCHLAFEN."
56830 DATA "HABEN WIR UNS NICHT SCHON BEGRUESST?"
56840 DATA "ICH SAG GERN AUCH EIN ZWEITES MAL(6SPACE)'GUTEN TAG'!"
56850 DATA &
56860 DATA ABSCHIED,30
56870 DATA "WIEDERSEHEN, +NAME.","ADE.", "TSCHUESSLE.","ADIEU."
56880 DATA "DU GEHST SCHON? NA DANN, AUF WIEDER-(3SPACE)SEHEN, +NAME."
56890 DATA &
56900 DATA AUSRUF,31
56910 DATA "NIMM'S NICHT SO SCHWER.","ES SIEHT SCHLIMMER AUS, ALS ES IST!"
56920 DATA "IST ALLES HALB SO SCHLIMM.","NUR MUT!"
56930 DATA "KEINE ANGST! DU WIRST ES SCHAFFEN.","NIMM'S NICHT SO TRAGISCH."
56940 DATA &
56950 DATA JA,32
56955 DATA "ACH SO.","NUN JA ...","GUT.","SCHOEN.","DANN BIN ICH BERUHIGT."
56960 DATA &
56965 DATA NEIN,33
56970 DATA "ACH SO?","NUN DENN...","NEIN?","SCHAD DRUM.","WIE DU WILLST!"
56975 DATA "DANN MAN NICHTS MACHEN.","NA JA, WAS SOLL'S ..."
56980 DATA &
56982 DATA DANK,39
56983 DATA "BITTE.","BITTE SEHR.","ABER FUER DICH TU ICH DOCH ALLES."
56984 DATA "BITTE SCHOEN.","ICH BIN DIR IMMER GERN BEHILFLICH."
56985 DATA &
56990 DATA ###
56995 :
57000 REM 5: WEISS NICHT
57010 DATA WEISSNICHT,35
57020 DATA "KEINE AHNUNG.","DARUEBER WEISS ICH LEIDER NICHTS."
57030 DATA "DA BIN ICH UEBERFRAGT.","DA MUSST DU SCHON EINEN EXPERTEN FRAGEN."
57040 DATA "ICH WEISS EINE GANZE MENGE, ABER DAS(3SPACE)LEIDER NICHT, +NAME."
57050 DATA "DAVON HAB ICH NIE WAS GEHOERT."
57060 DATA "TUT MIR LEID, WEISS ICH NICHT."
57065 DATA "WENN ICH DAS WUESSTE ...","DAS IST EINE GUTE FRAGE, BLOSS ...
57070 DATA &
57080 DATA ###
57090 :
57300 REM 6: VERSTEH NICHT
57310 DATA VERSTEHNICHT,35
57320 DATA "WIE BITTE?","WAS HAST DU GESAGT?"
57330 DATA "VERZEIHUNG, ICH HAB WOHL(RIGHT)GRAD(10SPACE)GESCHLAFEN."
57340 DATA "KOENNTEST DU DAS NOCH MAL IN ANDERN(4SPACE)WORTEN SAGEN?"
57350 DATA "ICH VERSTEH IMMER NUR BAHNHOF..."
57360 DATA "WAS MEINST DU DAMIT?"
57370 DATA "VIELLEICHT SOLLTEST DU DICH EINFACHER(2SPACE)AUSDRUECKEN."
57380 DATA "ICH VERSTEH NICHT GANZ - KOENNTEST DU(2SPACE)DAS NOCH EINMAL ANDER
      S SAGEN?
57390 DATA "WAS MEINST DU MIT: '+FRAGE'???"
57400 DATA "WAS HEISST '+FRAGE'???"
57410 DATA "'+FRAGE' - DAS VERSTEH ICH NICHT."
57420 DATA &
57430 DATA ###
57440 :
57500 REM 7: THEMA VERFEHLT
57510 DATA THEMAVERFEHLT,36
57520 DATA "HEISST DU UEBERHAUPT NOCH, WORUEBER DU REDEST?"
57530 DATA "HAST WOHL DAS THEMA VERGESSEN!?"
57540 DATA "WENN ICH MICH RECHT ERINNERE, WOLLTEN WIR UEBER COMAL REDEN!
57550 DATA "COMAL WAR UNSER THEMA, VERGESSEN?"
57560 DATA "WORUEBER REDEST DU EIGENTLICH?"
57570 DATA "VERGISS DAS THEMA NICHT!"
57580 DATA "COMAL IST UNSER THEMA!"
57590 DATA "BITTE LASS UNS BEIM THEMA BLEIBEN."
57600 DATA &
57610 DATA ###
57620 :
57800 REM 8: ABBRECHEN
57810 DATA ABBRUCH,37
57820 DATA "ICH GLAUBE, ES HAT KEINEN SINN, MIT DIR UEBER COMAL ZU REDEN.
57830 DATA "WIR VERSTEHEN UNS HEUTE OFFENBAR NICHT.
57840 DATA "ES HAT KEINEN ZWECK MIT UNS BEIDEN!
57850 DATA &
57860 DATA ###
57870 :
57900 REM ABSCHIEDSKOMMMENTAR
57905 DATA ENDE,38
57910 DATA "ICH HAB MICH GERN MIT DIR UNTERHALTEN, +NAME."
57915 DATA "HAT MICH GEFREUT, DICH KENNENZULERNEN."
57920 DATA "BESUCH MICH MAL WIEDER, +NAME."
57925 DATA "ICH HAETT MICH GERN NOCH LAENGER MIT(4SPACE)DIR UNTERHALTEN."
57930 DATA "SCHAD, DASS DU SCHON GEHST, +NAME."
57935 DATA &
57940 DATA ###
57950 :
57955 :
58900 REM FRAGEPRONOMEN
58910 DATA WER,WAS,WANN,WARUM,WIESO,WIE,WEM,WEN,WELCH,WESSEN,WOHER,WO,&
58980 :
58990 REM ENDE DATENMODUL
58995 :
58996 :
60000 REM INFORMATIONSMODUL
60005 REM ..................
60006 REM FRAGE- UND ANTWORTTYPEN
60007 :
60010 REM NR EINGABETYP  ANTWORTTYP
60015 REM ..............................
60020 REM 1: W-FRAGE     FAKTENANTWORT
60030 REM 2: JA/NEIN-FR  JA/NEIN-ANTW
60040 REM 3: ENTH 'DU'   'DU'-ANTWORT
60050 REM 4: AUSSAGE     AUSSAGE
60070 REM 5:             WEISS NICHT
60080 REM 6:             VERSTEH NICHT
60090 REM 7:             THEMA VERFEHLT
60100 REM 8:             ABBRUCH
```

Listing 1. »Comalchen« (Schluß)

Streifzüge durch die Grafikwelt (Teil 5)

In dieser Folge werden wir die Früchte zu ernten beginnen, deren Samen wir in der Vergangenen durch die Erklärung der Matrizen gesät haben. Anhand eines einfachen Beispiels sehen wir uns einige Transformationen auf dem Bildschirm an.

Erinnern Sie sich an die Definition einer Matrix aus der letzten Folge? Da hatten wir festgestellt, daß eine Matrix einfach eine geordnete rechteckige Darstellung von Elementen, beispielsweise von Zahlen ist. Nichts hindert uns also daran, auch die Koordinaten eines Punktes (siehe Bild 1) als eine Matrix aufzufassen.

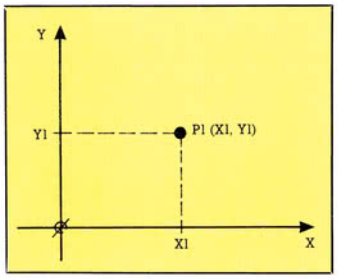

Bild 1. Der Punkt P1 im Koordinatensystem

Einem Punkt P1 (X1,Y1) entspricht also eine 1,2-Matrix, nämlich
(X1 Y1)

Nun können wir ausprobieren, was geschieht, wenn wir diese kleine Matrix allerlei Rechnereien aussetzen.

Skalieren

Was passiert, wenn wir die 1,2-Matrix eines Punktes mit einer 2,2-Matrix malnehmen? Bild 2 zeigt Ihnen ein Beispiel.

P1 ist unser Punkt, E die 2,2-Matrix. Das Ergebnis ist wieder eine 1,2-Matrix. Durch die Multiplikation ist P1 gar nicht verändert worden! Wir haben hier eine besondere Matrix gewählt, nämlich die sogenannte Einheitsmatrix, die auch als E bezeichnet wird. Eine Einheitsmatrix erkennt man immer an zwei Kriterien: Es ist eine quadratische Matrix, also eine, bei der die Anzahl der Zeilen und Spalten gleich ist, und sie enthält auf der Diagonalen von links oben nach rechts unten nur Einsen, während alle anderen Elemente gleich Null sind.

Sehen wir uns mal eine andere 2,2-Matrix an, die wir T1 nennen. Bild 3 zeigt Ihnen, was hier die Multiplikation erzeugt.

$$P1 = (X1\ Y1) \qquad E = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$1\ \ 0$$
$$0\ \ 1$$

$$X1 \quad Y1\ X1\ Y1$$
$$P1.E = (X1\ Y1)$$

Bild 2. Multiplikation von P1 mit der Matrix E

$$P1 = (X1\ Y1) \qquad T1 = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$$

$$2\ \ 0$$
$$0\ \ 1$$

$$X1 \quad Y12 \cdot X1 \quad Y1$$
$$P2 = P1 \cdot T1 = (2 \cdot X1\ Y1)$$

Bild 3. P1 wird mit der Matrix T1 malgenommen

P2 — der Punkt, der sich aus der Multiplikation P1*T1 ergibt — hat doppelt so große X-Koordinaten als P1. Setzen wir statt