

**KAPITEL 1****COMPUTERPROGRAMME****Was ist ein Programm?**

Die Frage, was ein Computerprogramm sei, ist nur schwer umfassend zu beantworten - aber es gibt eine einfache Näherung: Ein Programm ist eine Sammlung von Anweisungen an den Computer. Der ist nun - mit menschlichen Maßstäben gemessen - überaus dumm, d.h. er tut nichts von alleine (das ist wahrscheinlich ganz gut so), aber wenn er etwas tut, so tut er genau das, was den gegebenen Anweisungen entspricht. Diese Feststellung wird Sie, wie jede(n) von uns, noch manches Mal zur Verzweiflung bringen, wenn nämlich die Maschine wieder einmal nicht tut, was sie tun soll, sondern irgend etwas Unerwartetes. Dabei hat sie genau das getan, was den gegebenen Anweisungen entsprach, nur waren die offensichtlich nicht so gemeint. Ein Computer "weiß" aber leider nie, was gemeint war, im Gegensatz zu den netten Menschen in unserem Leben, die immer so tun, als ob sie es wüßten, der Computer nimmt alles äußerst wörtlich. Es kommt beim Schreiben von Programmen also auf Präzision und logische Klarheit an, in dieser Hinsicht ähnelt die Programmierung eines Computers der Mathematik. Als beruhigenden Ausgleich dafür kann man vielleicht die Tatsache ansehen, daß die Maschine niemals ungeduldig wird. Sie dürfen, ohne rot zu werden, all die Fehler auch machen, die jede(r) von uns gemacht hat und wieder machen wird.

Unter dem Gesichtspunkt der Verarbeitung durch den Computer ist ein Programm am einfachsten als Rezept zu verstehen. Man gibt zuerst die Zutaten an und erklärt dann, was jeweils in welcher Reihenfolge und Kombination mit den bereitgestellten Zutaten erfolgen soll. Nun gibt es Rezepte, die aus guten Zutaten schlechtes Essen machen, dann sind mit ziemlicher Sicherheit die Anweisungen für die Verarbeitung schlecht gewesen; im schlimmsten Fall kommt etwas Ungenießbares oder gar Gesundheitsschädliches heraus. Umgekehrt kann auch ein Meisterkoch aus falschen oder zu wenigen Zutaten kein großartiges Essen machen; fehlen die richtigen Gewürze, wird das fertige Gericht fade und nicht sehr schmackhaft.

Für die Programmierarbeit heißt das: es kommt sowohl darauf an, die richtigen Zutaten in ausreichender Menge und Vielfalt zu verwenden, als auch darauf, sie geeignet miteinander zu verbinden, um das Gewünschte zu erzeugen. Und ebenso wie man sich beim Kochen stets auf die gleichen drei Zutaten und die gleiche Verarbeitungsmethode beschränken kann, wodurch man - mit kleinen Variationen - im Wesentlichen auch stets das gleiche Gericht auf dem Teller hat, kann man sich beim Programmieren auf einige wenige Standardverfahren beschränken; man kommt erstaunlich weit damit, aber es ist irgendwo dürftig und fade: eben Imbissbude.

Versuchen Sie möglichst bald, sich die verschiedenen Konstruktionen und Methoden, die Component Pascal in gut überlegter Konzeption zur Verfügung stellt, anzueignen und sich die Unterschiede zwischen ähnlichen, aber nicht identischen Verfahren klarzumachen, so daß Sie diese später gezielt bei Ihren eigenen Programmen einsetzen können. Je besser Sie die grundlegenden Einzelheiten verstanden haben, desto leichter wird es Ihnen fallen, eigene Rezepte zu erfinden und sie so zu verändern, daß das Ergebnis wirklich Ihren Vorstellungen entspricht. Auch in dieser Hinsicht ist es bei Computerprogrammen wie beim Kochen, die Variationen oder auch Neuschöpfungen gelingen umso leichter und selbstverständlicher, je besser man das jeweilige "kleine EinMalEins" beherrscht.

Man kann den Weg von der Idee bis zum gelungenen Produkt, die strukturierte Zerlegung des Gesamtgedankens in Einzelteile und geeignete Verarbeitungsmethoden und schließlich das erneute Zusammenfügen der Teile als schöpferische Kreativität bezeichnen; in der Sprache der Informatiker klingt das ein klein wenig erschreckender, in ihr heißt der Vorgang "Algorithmisierung". Sie sollten sich aber von der Befremdlichkeit des Wortes nicht verunsichern lassen, gemeint bleibt die Umsetzung eines guten Gedankens in ein gutes, reproduzierbares Ergebnis und dabei kann Ihnen die Beherrschung der in Component Pascal steckenden Möglichkeiten helfen.

### 1.1. Component Pascal Programme

Laden Sie die Datei A.odc in die BlackBox Systemumgebung und lassen Sie das Modul kompilieren, d.h. in die Sprache des Computers übersetzen. Klicken Sie dazu im Menü *Dev* auf das Kommando *Compile*. Sie erhalten im Log-Fenster eine Meldung *compiling ...*. Das bedeutet, die Kompilation ist korrekt erfolgt, es hat keine Fehler(meldung) gegeben. Sie werden merken, daß dies eigentlich die Ausnahme darstellt. Nicht nur am Anfang Ihrer Programmierarbeit, sondern auch nach jahrelanger Erfahrung werden Sie immer wieder Fehlermeldungen erhalten, wenn Sie einen eigenen oder auch fremden Programmentwurf kompilieren lassen. Sie brauchen also keineswegs zu verzweifeln, wenn Ihre Arbeit nicht sofort zum beabsichtigten Ergebnis führt, Programmierprobleme gehören für Anfänger wie für die Erfahrensten zur täglichen Arbeit und oft bestehen die Schwierigkeiten nicht so sehr in der Erstellung eines Programms, sondern im Auffinden und Beseitigen von Fehlern.

Zurück zum Programm A.odc. Es gab bei der Kompilation keine Fehlermeldung, d.h. das Programm ist aus der Sicht des Compilers korrekt. Er kann es übersetzen, ohne daß etwas fehlt oder unverständlich ist. Versuchen Sie, irgendein Zeichen zwischen dem M des Wortes *MODULE* und dem Punkt am Ende der Zeile *END TutA*. wegzunehmen, lassen Sie das Programm erneut kompilieren und Sie werden sehen, daß Sie eine Fehlermeldung auf den Bildschirm bekommen. Anders gesagt: Sie haben das minimale Component Pascal Programm vor sich. Lediglich die beiden Zeilen der Überschrift - den Text in den (\* \*)-Klam-

mern - können Sie weglassen, es handelt sich um Kommentartext, der nur für Sie, den Leser, nicht für den Compiler gedacht ist. (Mehr dazu im Zusammenhang mit Programm 2 des nächsten Kapitels: *Komment.odc*).

Sie finden im Programm A.odc diejenigen Zutaten, die mindestens zu jedem Component Pascal Programm gehören. Der Grundbaustein eines Component Pascal Programms ist das Modul. Dieses beginnt mit dem sogenannten Modulkopf und endet mit einem Punkt. Der Modulkopf besteht aus dem Schlüsselwort *MODULE* und dem Modulnamen. Schlüsselwörter nennt man auch *Reservierte Bezeichner*, dies sind Zeichen oder Wörter mit einer festgelegten und unveränderbaren Bedeutung<sup>1</sup>. Dem Schlüsselwort *MODULE* folgt (mindestens) ein Leerzeichen, dann der Modulname und schließlich ein Semikolon. Component Pascal benötigt im Anschluss an das Wort *MODULE* stets einen Namen (hier *TutA*), der wenigstens aus einem Buchstaben bestehen muß. Anstelle von *TutA* könnten z.B. auch die Wörter *Opossum* oder *Mondkalb* stehen, irgendeine Zeichenfolge also, die als Modulname dient. Sie dürfen sich diese Zeichenfolge weitgehend frei aussuchen, Sie sollten dabei möglichst nur Buchstaben (und eventuell Ziffern) verwenden, auch wenn einige andere Zeichen erlaubt sind, die Sie auf der Tastatur finden. In jedem Fall aber muß das erste Zeichen des Namens ein Buchstabe sein.

Dem Modulnamen muß stets ein Semikolon folgen und innerhalb des Namens sind in keinem Fall Leerzeichen erlaubt. Leerzeichen haben in Component Pascal eine besondere Funktion, sie trennen Schlüsselwörter (z. B. das Wort *MODULE*) und andere Bezeichner (hier der Modulname *TutA*) voneinander. Über diese unbedingt notwendigen Leerzeichen hinaus können Sie an (fast) beliebiger Stelle eines Moduls weitere Leerzeichen in (fast) beliebiger Menge einfügen; dies ist eine sehr nützliche Eigenschaft, da sie Ihnen und mir erlaubt, Programme so zu schreiben, daß sie lesbar und übersichtlich sind.

Weiter finden Sie im Modul *TutA* das Schlüsselwort *END*. Dem *END* folgen die Wiederholung des Modulnamens (hier also *TutA*) und ein Punkt, der unbedingt benötigt wird; er signalisiert dem Compiler, daß an dieser Stelle das gesamte Modul zu Ende ist. Das Wort *END* allein reicht deshalb nicht, weil es als *END* anderer innerhalb des Moduls stehender Programmblöcke mehrfach vorkommen kann, also nicht in jedem Fall das definitive Modulende bedeutet.

Sie werden jetzt möglicherweise fragen, welchen Sinn das Modul *TutA* hat, da es nichts enthält. Module sind in Component Pascal die Einheiten der Kompilation. Man kann sie als Behälter ansehen, die notwendig sind, um die in ihnen (potentiell) vorhandenen Einzelteile zusammenzuhalten. Diese Einzelteile können nicht selbständig existieren, sie brauchen eine "Umhüllung", das umgebende Modul. Das Modul ist in Component Pascal der Grundbaustein, die (einzige) kompilationsfähige Einheit.

<sup>1</sup> Eine Liste mit Reservierten Bezeichnern finden Sie im Anhang C.

## 1.2. Kommandoprozeduren

Mit dem nächsten Programm, *Morgenstern.odc*, haben Sie ein Component Pascal Modul vor sich, das nicht nur kompilierbar ist wie das Modul *TutA*, sondern darüber hinaus eine - wenn auch einfache - Aktion ermöglicht. Aktionen können in Component Pascal entweder durch den Benutzer oder durch andere Programme bewirkt werden. Das zweite dieser Verfahren werden Sie später kennen lernen, das Modul *TutMorgenstern* macht Sie mit der Möglichkeit bekannt, als Benutzer eine Aktion auszulösen.

Grundsätzlich ist Component Pascal eine ereignisgesteuerte Programmiersprache, in der alle Aktionen von Prozeduren ausgeführt werden. Sie sehen in dem Modul *TutMorgenstern* eine solche Prozedur mit dem Namen *Start*. Eine Prozedur ist fast genauso aufgebaut wie ein Modul, sie besteht aus einem Prozedurkopf, der sich zusammensetzt aus dem Schlüsselwort *PROCEDURE* und dem Prozedurnamen. Das Ende der Prozedur wird wie bei einem Modul durch das Schlüsselwort *END* gefolgt vom Prozedurnamen gebildet, wobei diesem Namen kein Punkt, sondern ein Semikolon folgt. Zwischen diesen beiden Programmzeilen steht der Rumpf genannte Teil der Prozedur, der die eigentlichen Handlungsanweisungen enthält (genauer zu Prozeduren erfahren Sie im 5. Kapitel).

Im vorigen Abschnitt wurde erläutert, daß ein Modul eine Kompilationseinheit ist, eine Umhüllung. Damit es eine Interaktion zwischen dem Modul(inneren) und der Außenwelt geben kann, muß das Modul eine sogenannte Schnittstelle (Englisch: interface) haben, über die die außerhalb zur Verfügung stehenden Dienste exportiert werden. Zu exportierende Bezeichner werden in Component Pascal durch ein nachgestelltes Sternchen (\*) kenntlich gemacht. Das Modul *TutMorgenstern* exportiert somit einen einzigen Bezeichner, die Prozedur *Start*.

Exportierte Prozeduren wie *Start*, die im Prozedurkopf nur den Namen enthalten, werden in Component Pascal auch als Kommandoprozeduren oder kurz Kommandos bezeichnet, da sie außerhalb des Moduls aktiviert werden können. Die Aktivierung kann auf mehrere Weisen erfolgen, eine sehr einfache besteht darin, (nach erfolgreicher Kompilation des Moduls) das Kommando innerhalb der Component Pascal Umgebung an beliebiger Stelle in ein Textfenster zu schreiben, es zu markieren und im Menü *Dev* das Kommando *Execute* anzuklicken.

Mit Kommandos oder anderen von einem Modul exportierten Bezeichnern ist es ähnlich wie mit menschlichen Namen, zur vollständigen Identifikation eines Menschen wird nicht nur sein Vorname, sondern auch der Nachname gebraucht. In Component Pascal ist der Nachname eines exportierten Bezeichners der Name des exportierenden Moduls, allerdings ist die Reihenfolge der beiden Namensteile vertauscht und statt eines Leerzeichens steht zwischen den Namensteilen ein Punkt. Folglich lautet der vollständige Name des Kommandos *TutMorgenstern.Start* und muß zur Aktivierung in dieser Form benutzt werden. Schreiben Sie also das Kommando in das Log-Fenster oder unter den Programmtext, markieren Sie es und

lassen Sie es wie beschrieben ausführen, so erscheint als Ergebnis im Log-Fenster der Text des Gedichtes von Christian Morgenstern.

Sie sollten sich im Augenblick keine Gedanken um den übrigen Inhalt des Moduls machen, die Einzelheiten werden im 2. Kapitel im Zusammenhang mit dem Programm *Text.odc* erklärt, wichtig ist im Augenblick lediglich, daß Sie das Modul erfolgreich kompiliert und das Kommando *TutMorgenstern.Start* mit dem gewünschten Ergebnis aktiviert haben.

## 1.3. Der Compiler

Laden Sie das Programm *Hallo.odc* und kompilieren Sie das Modul. Im Log-Fenster erhalten Sie die Meldung *compiling 2 errors detected* und im Quelltext erscheinen zwei Fehlermarken an den Positionen der Fehler. Doppel-Klicken Sie auf die erste dieser Fehlermarken, entfaltet sie sich zu dem Text *undeclared identifier*. Ändern Sie den dem Compiler nicht bekannten Bezeichner *OutString*, indem Sie einen separierenden Punkt einfügen (*Out.String*; genauer zu diesem Bezeichner erfahren Sie im nächsten Kapitel im Zusammenhang mit dem Programm *Text.odc*). Doppel-Klicken Sie auf die zweite Fehlermarke, es erscheint der Text *identifier expected*. Fügen Sie ein Leerzeichen und den fehlenden Modulnamen (*TutHallo*) vor dem Punkt ein. Lassen Sie das Programm neu kompilieren. Im Log-Fenster erscheint die Meldung *compiling 28 0*, das Modul wurde übersetzt, es ist syntaktisch fehlerfrei und es sind zwei Dateien erzeugt worden, *Hallo.ocf* mit dem lauffähigen Programmcode im Unterverzeichnis *Tut/Code* des Arbeitsverzeichnisses sowie die Symboldatei *Hallo.osf* mit der Beschreibung der vom Modul *TutHallo* exportierten Bezeichner im Unterverzeichnis *Tut/Sym*.

Markieren Sie jetzt das unterhalb des Moduls stehende Kommando *TutHallo.Welt* und lassen Sie es ausführen (Menü *Dev* → *Execute*). Sie erhalten im Log-Fenster die Meldung *command error: command Welt not found in TutHallo*, das Kommando *Welt* ist vom Modul *TutHallo* nicht exportiert. Fügen Sie im Prozedurkopf *PROCEDURE Welt*; zwischen dem Prozedurnamen *Welt* und dem Semikolon die Exportmarke \* ein und lassen Sie das Modul erneut kompilieren, im Log-Fenster erscheint *compiling Welt is new in symbol file 32 0*, das Kommando *TutHallo.Welt* ist exportiert.

Lassen Sie nun das (markierte) Kommando *TutHallo.Welt* erneut ausführen, so erhalten Sie dieselbe Meldung *command error: command Welt not found in TutHallo* wie zuvor. Dies liegt daran, daß sich das ursprüngliche Modul (ohne exportiertes Kommando *Welt*) weiterhin im Arbeitsspeicher befindet. Um das neu kompilierte Modul zu laden, müssen Sie das alte Modul aus dem Speicher entfernen (*Dev* → *Unload*). Im Log-Fenster erscheint der Text *TutHallo unloaded*. Lassen Sie das Kommando *TutHallo.Welt* danach erneut ausführen, erscheint im Log-Fenster der Text *Hallo Welt*, das neu kompilierte Modul ist wieder geladen und das Kommando *TutHallo.Welt* ist ausgeführt worden.