Die PASCAL-Fibel

Strukturierte Programmierung mit Pascal
Lehr- und Arbeitsbuch für Anfänger
(Für alle gängigen Pascal-Systeme)

Von
Dr. Peter Witschital
und
Dr. Thomas Kühme
überarbeitet und erweitert von
Dipl.-Inform. Bettina Meiners

Fünfte, überarbeitete und erweiterte Auflage

R. Oldenbourg Verlag München Wien

Dr. Peter Witschital
Am Treiberweg 26
D-85630 Grasbrunn

Dr. Thomas Kühme
127 Lake Meadow Drive
Johnson City, TN 37615
U.S.A.

Dipl.-Inform. Bettina Meiners, geb. Benzel
Institut für Programmiersprachen und Informationssysteme
Abteilung Programmiersprachen
Technische Universität Braunschweig
Gaußstr. 11
D-38106 Braunschweig

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Witschital, Peter
Die Pascal-Fibel : strukturierte Programmierung mit Pascal ; Lehr- und Arbeitsbuch für Anfänger ; (für alle gängigen Pascal-Systeme) / von Peter Witschital und Thomas Kühme. - 5., überarb. und erw. Aufl. / überarb. und erw. von Bettina Meiners. - München ; Wien : Oldenbourg, 1996
ISBN 3-486-23262-2
NE: Kühme, Thomas:; Meiners, Bettina [Bearb.]

Gesamtherstellung: R. Oldenbourg Graphische Betriebe GmbH, München

ISBN 3-486-23262-2

Vorwort

Eine Fibel ist ein Lesebuch, ein Elementarlehrbuch, ein Lernbuch zur elementaren Einführung in ein Sachgebiet — so heißt es bei Duden und Brockhaus.

Die Pascal-Fibel ist einerseits ein Lehrbuch für Programmieranfänger und soll den Leser am Beispiel der weit verbreiteten Programmiersprache Pascal in die Grundzüge des Programmierens einführen. Nicht die Darstellung des vollen Pascal-Sprachumfangs steht im Vordergrund, sondern die leicht verständliche und ausführliche Erläuterung der wichtigsten Konzepte der Programmiersprache Pascal. Dabei stehen diejenigen Konzepte im Vordergrund, die man auch in anderen Programmiersprachen wiederfindet. Vorkenntnisse aus Informatik, Mathematik oder anderen Bereichen sind nicht erforderlich.

Die Pascal-Fibel ist aber auch ein Lesebuch. Neue Sprachelemente werden möglichst erst dann vorgestellt, wenn eine Veranlassung dazu gegeben ist. Der Leser wird nicht zu früh mit Dingen belastet, deren Sinn und Zweck er noch nicht verstehen kann. Die Pascal-Fibel kann daher gut Seite für Seite durchgelesen werden. Das ist wichtig, denn die Fibel ist für das Selbststudium gedacht. Der Kritik, die Fibel sei daher ein weniger gutes Nachschlagewerk, begegnen wir mit einem ausführlichen Stichwortverzeichnis.

Die Einführung in das Programmieren erfolgt anhand vieler ausführlicher Beispiele. Im Anschluß an die Lektionen und Unterlektionen werden Kontrollaufgaben gestellt, deren Lösungen man im Anhang findet. Schon nach kurzer Zeit ist der Leser in der Lage, kleine, aber deshalb nicht weniger anspruchsvolle Übungsaufgaben zu lösen und seinen Lernerfolg daran zu messen.

Die Pascal-Fibel sowie die gleichzeitig erscheinende FORTRAN-Fibel entstanden als Unterlagen zu einem Programmierkurs für Studenten aller Fachrichtungen und Semester an der Technischen Universität Braunschweig. Der Programmierkurs wird auf Arbeitsplatzrechnern mit dem Pascal-System PolyPascal durchgeführt, das dem weit verbreiteten Turbo-Pascal ähnelt.

In der Pascal-Fibel wird sowohl Standard-Pascal als auch PolyPascal beschrieben. Auf Unterschiede wird jeweils im Text hingewiesen. An einigen Stellen, insbesondere bei der Dateibearbeitung, geht die Pascal-Fibel intensiver auf die Besonderheiten von PolyPascal ein. Die Hinweise für PolyPascal gelten im allgemeinen auch für Turbo-Pascal.

Bei allen Kursteilnehmern und -betreuern des Programmierkurses möchten wir uns herzlich für die vielen wertvollen Hinweise und Anregungen bedanken.

Unser Dank gilt auch und vor allem Herrn Prof. Dr. Günther Stiege, der den Anstoß zur Entstehung der Fibeln gab und uns bei diesem Vorhaben umfassende Unterstützung gewährte. Besonderen Dank möchten wir Herrn Karsten Luck und Herrn Arnd Gerns für die Erstellung der Druckvorlage unter Verwendung des Programms TEX aussprechen. Nicht zuletzt danken wir unseren Angehörigen für das uns entgegengebrachte Verständnis und die Hilfe beim Korrekturlesen. Beschließen möchten wir die Danksagungen mit einem herzlichen Dankeschön an den Oldenbourg Verlag, der bereitwillig auf unsere Wünsche bei der Gestaltung der Fibel einging.

Den Lesern der Pascal-Fibel wünschen wir viel Erfolg beim Programmieren!

Braunschweig, im Frühjahr 1987

Peter Witschital
Thomas Kühme

Vorwort zur 3. Auflage

Die Pascal-Fibel geht nun, zwei Jahre nach Erscheinen der Erstauflage, bereits in die dritte Auflage. Das "Fibel-Konzept" hat sich erfolgreich bewährt und bedarf daher noch keiner Überarbeitung. In die vorliegende Neuauflage wurden einige Korrekturen und Ergänzungen aufgenommen. Die Übungsaufgaben im Anhang C wurden teilweise durch neue, noch interessantere Aufgaben ersetzt. Mit dem neuen Kurzindex zum Herausklappen wurde eine oft geäußerte Anregung verwirklicht. In Verbindung mit der bewährten Spiralbindung kann die Fibel – über ihren Lehrbuch-Charakter hinaus – nun noch besser als Arbeitsbuch direkt am Rechner verwendet werden.

Wir danken allen, die durch ihre Kritik, Anregungen und Mitwirkung zum guten Gelingen dieser Auflage beigetragen haben.

Braunschweig, im Frühjahr 1989

Peter Witschital
Thomas Kühme

Vorwort zur 4. Auflage

Wir freuen uns, hiermit eine vierte, sorgfältig durchgesehene Auflage der Pascal-Fibel präsentieren zu können.

Die durchweg positive Resonanz, die in den Briefen unserer Leser zum Ausdruck kommt, belegt, daß mit dem Fibel-Konzept der Einstieg in das Programmieren mit Pascal ganz wesentlich erleichtert wird. Wir freuen uns über diesen Erfolg und wünschen allen künftigen Lesern ein gutes Gelingen!

Peter Witschital
Thomas Kühme

Vorwort zur 5. Auflage

Eine Fibel ist ein Lesebuch, ein Elementarlehrbuch, ein Lernbuch zur elementaren Einführung in ein Sachgebiet – so heißt es bei Duden und Brockhaus.

Die Pascal-Fibel ist einerseits ein Lehrbuch für Programmieranfänger und soll den Leser am Beispiel der weit verbreiteten Programmiersprache Pascal in die Grundzüge des Programmierens einführen. Nicht die Darstellung des vollen Pascal-Sprachumfangs steht im Vordergrund, sondern die leicht verständliche und ausführliche Erläuterung der wichtigsten Konzepte der Programmiersprache Pascal. Dabei stehen diejenigen Konzepte im Vordergrund, die man auch in anderen Programmiersprachen wiederfindet. Vorkenntnisse aus Informatik, Mathematik oder anderen Bereichen sind nicht erforderlich.

Die Pascal-Fibel ist aber auch ein Lesebuch. Neue Sprachelemente werden möglichst erst dann vorgestellt, wenn eine Veranlassung dazu gegeben ist. Der Leser wird nicht zu früh mit Dingen belastet, deren Sinn und Zweck er noch nicht verstehen kann. Die Pascal-Fibel kann daher gut Seite für Seite durchgelesen werden. Das ist wichtig, denn die Fibel ist für das Selbststudium gedacht. Der Kritik, die Fibel sei daher ein weniger gutes Nachschlagewerk, begegnen wir mit einem ausführlichen Stichwortverzeichnis und einem Kurzindex zum schnellen Auffinden aller wichtigen Stellen.

Die Einführung in das Programmieren erfolgt anhand vieler ausführlicher Beispiele. Im Anschluß an die Lektionen und Unterlektionen werden Kontrollaufgaben gestellt, deren Lösungen man im Anhang findet. Schon nach kurzer Zeit ist der Leser in der Lage, kleine, aber deshalb nicht weniger anspruchsvolle Übungsaufgaben zu lösen und seinen Lernerfolg daran zu messen.

Die Pascal-Fibel sowie die gleichzeitig erschienene FORTRAN-Fibel entstanden 1987 als Unterlagen zu einem Programmierkurs für Studenten aller Fachrichtungen und Semester an der Technischen Universität Braunschweig. In den ersten vier Auflagen dieser Fibel wurde neben Standard-Pascal auch der Pascal-Dialekt PolyPascal beschrieben, der damals in den praktischen Übungen zum Programmierkurs verwendet wurde.

In nunmehr acht Jahren hat sich die Fibel nicht nur im Unterricht mit mehr als 3.000 Studenten bewährt. Der Aufbau der Fibel und der lockere Stil sind sehr gut angekommen und ermöglichen nach wie vor einen leichten Einstieg in

das Programmieren. Inzwischen haben sich die Voraussetzungen dieses Programmierkurses verändert. Es wird nicht mehr mit PolyPascal, sondern zur Zeit mit XL Pascal gearbeitet. Immer mehr Studenten arbeiten am eigenen Rechner zu Hause mit Turbo-Pascal. Um die Fibel noch universeller und unabhängiger von bestimmten Programmiersystemen zu gestalten, wurde sie komplett überarbeitet.

In der Fibel wird jetzt grundsätzlich Standard-Pascal beschrieben. An den Stellen, an denen Standard-Pascal nicht als ausreichende Grundlage genügt, wird auf diese Tatsache ausdrücklich hingewiesen und Turbo-Pascal zugrunde gelegt. In diesem Zusammenhang beschreibt jetzt die Lektion 'Dateibearbeitung' die Handhabung von Dateien mit Direktzugriff unter Turbo-Pascal.

Die Fibel wurde außerdem um einen 'Anhang D' erweitert, der Abweichungen der Pascal-Versionen Turbo-Pascal, PolyPascal und XL Pascal von dem beschriebenen Standard-Pascal erläutert. Er soll dem Leser helfen, die in der Fibel erlernte Programmiersprache auch mit einem anderen Pascal-System anwenden zu können.

Überarbeitet und erneuert wurde weiterhin die Lektion 'Dynamische Variablen und Zeigertypen'. Es wird jetzt ausführlicher auf die rechnerinterne Darstellung und Verwaltung von dynamischen Variablen eingegangen.

Bei allen Kursteilnehmern und -betreuern des Programmierkurses möchten wir uns herzlich für die vielen wertvollen Hinweise und Anregungen bedanken.

Unser Dank gilt auch und vor allem Herrn Prof. Dr. Günther Stiege, der den Anstoß zur Entstehung der Fibeln gab und uns bei diesem Vorhaben umfassende Unterstützung gewährte. Besonderen Dank möchten wir Herrn Karsten Luck und Herrn Arnd Gerns für die Erstellung der Druckvorlage unter Verwendung des Programms TEX aussprechen. Im Zusammenhang mit der Überarbeitung der Fibel für die fünfte Auflage danken wir Herrn Martin Neitzel für seine Unterstützung in technischen Fragen und seine wertvollen Anregungen zur Neugestaltung von Lektion 14.2.

Nicht zuletzt danken wir unseren Angehörigen für das uns entgegengebrachte Verständnis und die Hilfe beim Korrekturlesen. Beschließen möchten wir die Danksagungen mit einem herzlichen Dankeschön an den Oldenbourg Verlag, der bereitwillig auf unsere Wünsche bei der Gestaltung der Fibel einging.

Den Lesern der Pascal-Fibel wünschen wir viel Erfolg beim Programmieren!

Braunschweig, im Sommer 1995

Peter Witschital
Thomas Kühme
Bettina Meiners

Inhaltsverzeichnis

Lektionen:

Lektion 1

Hinweise zum Lesen der Fibel

Die Pascal-Fibel ist für das Selbststudium konzipiert. Sie ist in Lektionen aufgeteilt, die aufeinander aufbauen und die Sie daher in der vorgegebenen Reihenfolge durchgehen sollten.

Die Aufteilung des Lehrstoffs auf die Lektionen erfolgte nach thematischen Gesichtspunkten. Die einzelnen Lektionen haben daher recht unterschiedlichen Umfang. Insofern ist der Begriff "Lektion" in der Pascal-Fibel nicht synonym zu "Lerneinheit". Die umfangreichen Lektionen sind deshalb weiter untergliedert und so in mundgerechte (hirngerechte) Happen aufgeteilt.

Bevor Sie sich das erste Mal an den Rechner setzen, sollten Sie Lektion 2 lesen. In Lektion 3 werden Sie aufgefordert, sich mit dem Ihnen zur Verfügung stehenden Rechnersystem vertraut zu machen. Während Sie im Laufe der weiteren Lektionen zum (zur) Pascal-Experten(in) werden, sollte der Rechner Ihr ständiger Begleiter bleiben.

In den ersten Lektionen werden Sie dazu angehalten, die Beispielprogramme in den Rechner einzugeben, auszuprobieren und mit ihnen zu experimentieren. Später sollten Sie mit selbst ausgewählten Beispielen entsprechend verfahren.

Überspringen Sie nichts und versuchen Sie, alle Kontrollaufgaben zu lösen, die im Anschluß an die Lektionen bzw. Unterlektionen gestellt werden. Für fast alle Kontrollaufgaben sind Lösungen im Anhang angegeben. Sehen Sie dort aber erst nach, wenn Sie sich ernsthaft mit einer Aufgabe befaßt haben.

Hin und wieder werden Sie im Anschluß an eine Lektion aufgefordert, eine Übungsaufgabe zu bearbeiten. Die Übungsaufgaben sind Aufgaben ohne angegebene Lösungen. Sie geben Ihnen Gelegenheit, die Anwendung des Wissens, das Sie bis dahin erworben haben, an einem praktischen Beispiel zu üben. Wenn Sie alle Übungsaufgaben erfolgreich bearbeitet haben, können Sie zu Recht von sich behaupten, Pascal programmieren zu können. Sie finden die Übungsaufgaben im Anhang C der Pascal-Fibel.

Programmtexte sowie Ein- und Ausgaben von Programmen werden in der Fibel durch einen stilisierten Bildschirm hervorgehoben und zusätzlich durch einen `speziellen Schrifttyp`, auch im fortlaufenden Text, gekennzeichnet:

```
Dies ist ein Text auf dem Bildschirm.
```

Zur allgemeinen Darstellung eines Programmschemas wird in der Fibel die folgende Notation verwendet:

`if` <Bedingung> `then` <Anweisung> `else` <Anweisung>

Die <Begriffe in Spitzklammern> sind als Platzhalter zu verstehen, die noch durch einen geeigneten Programmtext ersetzt werden müssen.

Die Sprachgrundlage der Beschreibungen in der Pascal-Fibel bildet Standard-Pascal nach ISO-Norm (ISO 7185). In einigen Fällen reicht diese Norm als Grundlage nicht aus. Die tatsächlichen Anwendungen sind hier sehr verschieden. Es wird deshalb an diesen Stellen die verbreitete Version von Turbo-Pascal als Spezialfall beschrieben. Die wichtigsten Abweichungen einiger Pascal-Versionen von dieser Norm sind im Anhang D zusammengefasst.

Doch nun genug der Vorrede, los geht's!

Lektion 2

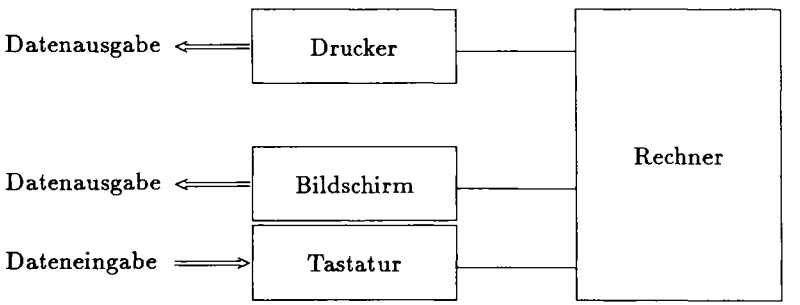
Grundbegriffe der Programmierung

2.1 Daten, Programme, Programmiersprachen

Dieser Abschnitt richtet sich an alle diejenigen, die sich erstmalig mit der Programmierung eines Rechners beschäftigen. Die wichtigsten Begriffe dieses Problemkreises werden kurz erläutert.

Der Begriff **Rechner** (engl.: Computer) ist eigentlich zu speziell gefaßt. Denn ein Rechner kann nicht nur mit Zahlen rechnen, sondern auch **Daten** ganz anderer Art verarbeiten, zum Beispiel Texte.

Die Verarbeitung von Daten findet immer in drei Schritten statt. Der erste Schritt ist die **Dateneingabe**, die zum Beispiel über eine Tastatur erfolgen kann. Als zweites folgt die eigentliche **Verarbeitung** der Daten im Rechner und der dritte Schritt ist die **Ausgabe** der Ergebnisse, zum Beispiel auf einen Bildschirm oder einen Drucker.



Wenn wir mit dem Rechner etwas anfangen wollen, müssen wir ihn irgendwie dazu bringen, das zu tun, was wir von ihm wollen. Dazu müssen wir ihm ein **Programm** geben. Ein Programm enthält eine Reihe von **Anweisungen** an den Rechner, die jeden Verarbeitungsschritt genau festlegen. Erst, wenn dem Rechner ein Programm als Ganzes vorliegt, kann er es auf Wunsch beliebig oft und mit verschiedenen Daten ausführen.

Nehmen wir an, wir wollen den Rechner zur Addition zweier beliebiger ganzer Zahlen einsetzen. Die Zahlen sollen über die Tastatur eingegeben werden, und das Ergebnis soll auf dem Bildschirm erscheinen.

"Beliebige Zahlen" bedeutet dabei nicht, daß wir uns zwei Zahlen aussuchen können, sondern daß wir beim Schreiben des Programms noch nicht wissen, welche beiden Zahlen später mit dem Rechner addiert werden sollen.

Bevor wir ein Programm erstellen können, müssen wir uns überlegen, welche Daten wir dem Rechner geben wollen, was er damit machen soll und was wir als Ergebnis erhalten wollen. Die Eingabedaten für unser Beispiel sind zwei Zahlen, das einzige Ausgabedatum deren Summe. Die eigentliche Verarbeitung der Daten besteht lediglich aus der Addition der beiden Zahlen. Wir wollen nun ein Programm aufschreiben, das die notwendigen Schritte enthält und zunächst in unserer Umgangssprache formuliert ist.

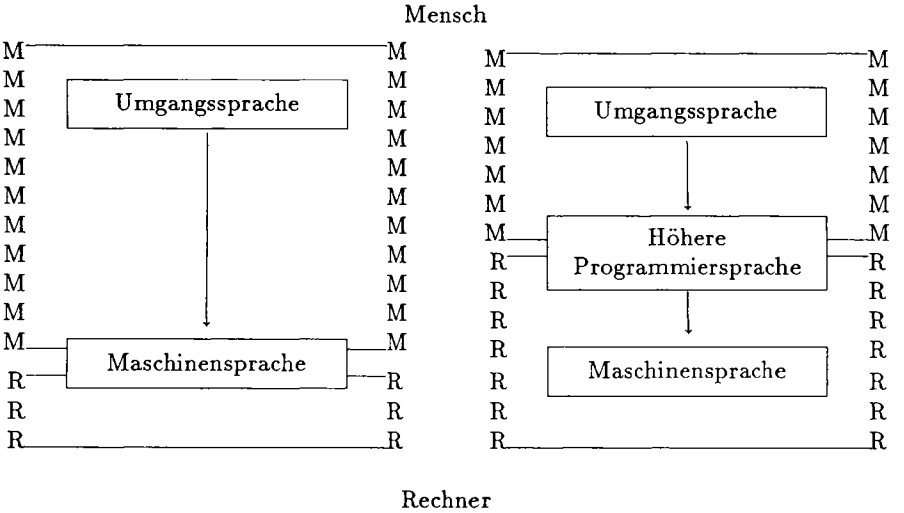
Additionsprogramm:

1. Lies die beiden zu addierenden Zahlen von der Tastatur ein !
2. Addiere die beiden Zahlen!
3. Gib die Summe auf dem Bildschirm aus !

Leider ist der Rechner ziemlich dumm. Wenn wir unser Programm in dieser Form in den Rechner eingeben, wird er sich beharrlich weigern, auch nur im entferntesten zu verstehen, was wir von ihm wollen. Der Rechner versteht nämlich kein Deutsch, sondern nur seine spezielle **Maschinensprache**.

Man könnte nun darangehen, die umgangssprachlichen Anweisungen in die sogenannten **Maschinenbefehle** zu übertragen, diese in den Rechner einzugeben und ihn dann ausführen zu lassen (im Bild links).

Aber die Maschinensprache kennt nur zwei "Buchstaben" (0 und 1) und läßt sich sehr schlecht aussprechen. Außerdem braucht man für eine Anweisung der Umgangssprache unter Umständen sehr viele Wörter in der Maschinensprache (wegen der wenigen Buchstaben), und schließlich hat fast jeder Rechner seine eigene Maschinensprache, die nur er und kein anderer Rechner versteht. Maschinensprache ist also für den Menschen sehr schwer zu lernen.



Um die Verständigung zwischen Mensch und Rechner zu erleichtern, wurden **höhere Programmiersprachen** entwickelt. Diese basieren auf der grundsätzlichen Funktionsweise von Rechnern, sind aber nicht auf einen speziellen Rechnertyp abgestimmt. Ihre Anweisungen orientieren sich mehr an der Umgangssprache und sind somit leichter zu verstehen und zu erlernen. Gegenüber den Anweisungen der Umgangssprache haben die Anweisungen von höheren Programmiersprachen den Vorteil, daß sie nach Form und Bedeutung einer strengen Festlegung genügen, um ganz präzise formulieren zu können, was man vom Rechner will. Bei komplizierteren Problemstellungen ist dies in der Umgangssprache nicht immer möglich.

Allerdings sind nun zwei Umsetzungen des Programmtextes notwendig (im Bild rechts). Die eine, von der Umgangssprache in die höhere Programmiersprache, führt der Programmierer aus. Diese Umsetzung wird **Codierung** eines Programms genannt, und das Ergebnis ist das **Quellprogramm** in der jeweiligen Programmiersprache.

Die noch unvollständig codierte Fassung unseres Beispielprogramms würde in einigen bekannten höheren Programmiersprachen in etwa so aussehen:

BASIC:
```
10 INPUT ZAHL1, ZAHL2
20 SUMME = ZAHL1 + ZAHL2
30 PRINT SUMME
40 END
```

FORTRAN 77:
```
READ *, ZAHL1, ZAHL2
SUMME = ZAHL1 + ZAHL2
PRINT *, SUMME
END
```

PASCAL:
```
BEGIN
READ (ZAHL1, ZAHL2);
SUMME := ZAHL1 + ZAHL2;
WRITELN (SUMME)
END.
```

Die andere Umsetzung, von der höheren Programmiersprache in die Maschinensprache, kann der Rechner übernehmen, wenn ihm ein **Übersetzer** (das ist auch ein spezielles Programm) zur Verfügung steht, der die höhere Programmiersprache in seine Maschinensprache übersetzen kann. Bei so einer Übersetzung wird jede Anweisung des Programms in der höheren Programmiersprache im allgemeinen in mehrere Maschinenbefehle übersetzt. Das entstehende Maschinenprogramm wird als Ergebnis des Übersetzungsvorgangs **Objektprogramm** genannt. Das Objektprogramm kann vom Rechner direkt ausgeführt werden.

Sehen wir uns den Ablauf der Programmierung noch einmal zusammenfassend an. Ausgangspunkt ist die Problemstellung. Es folgen dann:

- Formulierung eines Programmtextes in der Umgangssprache
- Codierung des Programms in der höheren Programmiersprache
- Eingabe des Quellprogramms in den Rechner
- Übersetzung des Quellprogramms in ein Maschinenprogramm durch den Rechner (genauer: durch das Übersetzerprogramm im Rechner)
- Ausführung des Objektprogramms durch den Rechner (beliebig oft)

Wie läuft nun die Ausführung eines Programms ab ? Wir nehmen an, wir hätten die vorstehenden Schritte für unser Beispiel, die Addition zweier Zahlen, durchgeführt und hätten den Rechner bereits dazu veranlaßt, die Ausführung unseres Programms zu starten.

Der Rechner führt die erste Anweisung aus, die ihm sagt, daß er zwei Zahlen von der Tastatur einlesen soll. Er wartet so lange, bis wir tatsächlich zwei Zahlen über die Tastatur eingeben, zum Beispiel `3` und `5`, getrennt durch einen Zwischenraum. Erst, wenn der Rechner unsere Eingabe erhalten hat, fährt er mit der Ausführung des Programms fort und addiert die beiden Zahlen gemäß der nächsten Anweisung. Mit der dritten Anweisung erfolgt die Ausgabe der Summe auf dem Bildschirm, in diesem Fall `8`. Danach wird das Programm beendet.

Wir können das Programm noch mehrfach anstarten, d.h., die Ausführung des Objektprogramms veranlassen. Jedesmal können andere Daten eingegeben werden. In einem zweiten Programmlauf würde zum Beispiel die Eingabe der Zahlen `789` und `310` zu dem Ergebnis `1099` auf dem Bildschirm führen.

Kontrollaufgaben

K.2.1 Zwei Zahlen sollen addiert werden, und die Summe soll mit einer dritten multipliziert werden. Die Zahlen sollen über die Tastatur eingegeben werden, und das Ergebnis soll auf dem Bildschirm erscheinen. Schreiben Sie ein Programm in der Umgangssprache !

K.2.2 Wie unterscheiden sich Maschinensprachen von höheren Programmiersprachen? Wo liegen die Vorteile der höheren Programmiersprachen?

K.2.3 Nennen Sie ein paar höhere Programmiersprachen!

K.2.4 Was versteht man unter der Codierung eines Programms ?

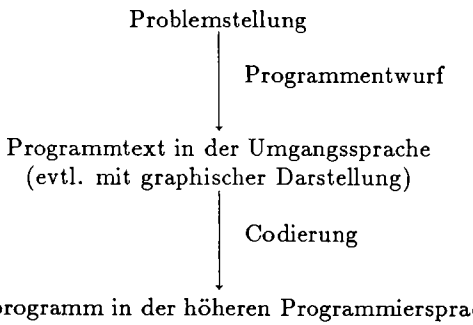
K.2.5 Was ist ein Quellprogramm und was ein Objektprogramm ?

K.2.6 Wer übersetzt ein Quellprogramm in ein Objektprogramm ?

2.2 Vom Problem zum Programm

Im vorhergehenden Abschnitt wurde der Ablauf der Programmierung in einer höheren Programmiersprache schematisch beschrieben. Dabei war für das einfache Beispiel "Additionsprogramm" die Formulierung eines Programmtextes in der Umgangssprache anschaulich klar und ausgehend von der Problemstellung ohne Zwischenschritte möglich. Im allgemeinen ist dieser Schritt, nämlich der eigentliche Programmentwurf, jedoch wesentlich aufwendiger und kann als Kernstück jeder Programmentwicklung angesehen werden. Die Vorgehensweise beim Programmentwurf ist Gegenstand dieses Abschnitts. Graphische Darstellungen der Programmabläufe und -strukturen sind nützliche Hilfsmittel und werden kurz erläutert.

Bevor wir auf den Programmentwurf eingehen, wollen wir diesen Vorgang nochmals in den Gesamtablauf der Programmierung einordnen:



Eine Problemstellung der Datenverarbeitung ist gegeben durch die Beschreibung der in das Problem eingehenden Daten und der gewünschten Ergebnisse.

Der Programmtext muß für das jeweilige Problem einen Lösungsweg darstellen, der eindeutig festlegt, wie die Verarbeitung zu erfolgen hat, um aus den eingehenden Daten die gewünschten Ergebnisse zu erhalten. Ein solcher Lösungsweg wird **Algorithmus** genannt.

Der Lösungsweg für das Problem, aus zwei Zahlen die Summe zu ermitteln, besteht, wie wir gesehen haben, einfach darin, die beiden Zahlen zu addieren. Als Beispiel für einen etwas umfangreicheren Lösungsweg wollen wir folgendes Problem betrachten.

Es seien drei Zahlen gegeben, und es soll der größte Abstand zwischen je zwei dieser Zahlen bestimmt werden. Ein- und Ausgabe sollen über das Bildschirmgerät erfolgen.

Der folgende Programmtext stellt einen möglichen Algorithmus zur Lösung dieser Aufgabe mit dem Rechner dar:

1. Lies die drei Zahlen über die Tastatur ein !
2. Bestimme die größte der drei Zahlen !
3. Bestimme die kleinste der drei Zahlen !
4. Ermittle den gesuchten Abstand durch Subtraktion der kleinsten von der größten Zahl !
5. Gib den Abstand als Ergebnis aus !

Jede der Anweisungen 1. bis 5. stellt ein Teilproblem dar, das unter Umständen wieder durch einen Algorithmus gelöst werden muß. Die Anweisungen müssen so lange "verfeinert" werden, bis sie in der gewählten Programmiersprache codiert werden können.

Für die Programmiersprache FORTRAN wäre der Programmentwurf für das Beispiel bereits abgeschlossen, denn in FORTRAN kann man Anweisungen formulieren, die das Maximum bzw. Minimum von drei Zahlen bestimmen. Das Programm ließe sich in FORTRAN direkt codieren. Die Anweisungen 2. und 3. müßten hingegen für eine Programmierung in Pascal weiter zerlegt werden, denn es gibt in dieser Sprache keine Anweisungen mit entsprechender Bedeutung wie in FORTRAN. Die größte und die kleinste Zahl könnten stattdessen zum Beispiel folgendermaßen gefunden werden:

2a. Bestimme die größere von der ersten und der zweiten Zahl !

2b. Bestimme die größte Zahl als Maximum der größeren der beiden und der dritten Zahl !

3a. Bestimme die kleinere von der ersten und der zweiten Zahl !

3b. Bestimme die kleinste Zahl als Minimum der kleineren der beiden und der dritten Zahl !

Für diese Anweisungen gibt es entsprechende in Pascal, so daß der Programmentwurf hiermit fertig ist und das Programm codiert werden kann.

Im allgemeinen empfiehlt es sich, den Lösungsweg zunächst möglichst grob aufzuschreiben und eine **schrittweise Verfeinerung** so lange wie beschrieben durchzuführen, bis jedes Teilproblem einer Anweisung oder einer kurzen Anweisungsfolge in der zu verwendenden höheren Programmiersprache entspricht.

Neben der Herleitung eines eindeutigen Lösungsweges muß beim Programmentwurf festgelegt werden, welche Daten verarbeitet werden sollen. Dazu gehören nicht nur die Ein- und Ausgabedaten, sondern auch jegliche Zwischenergebnisse.

Für das obige Beispiel ergibt sich als Aufstellung der zu verarbeitenden Daten folgendes:

- drei Zahlen (Eingabe)
- die größte Zahl (Zwischenergebnis)
- die kleinste Zahl (Zwischenergebnis)
- der Abstand (Endergebnis bzw. Ausgabe)

Je nach verwendeter Programmiersprache kann oder muß diese Aufstellung ebenfalls codiert und zum Lösungsweg hinzugefügt werden.

Mit Hilfe graphischer Darstellungen kann man Programme übersichtlicher darstellen und sich dadurch die Entwurfsarbeit erleichtern. Gebräuchliche Darstellungsformen sind **Programmablaufpläne** (Flußdiagramme) und **Struktogramme** (Nassi-Shneiderman-Diagramme).

Struktogramme sind Blockdiagramme, die die Struktur der Programme hervorheben, aber auch den Programmablauf erkennen lassen.

Mögliche Strukturen sind neben der rein sequentiellen Abarbeitung der Anweisungen eines Programms zum Beispiel die mehrfache Wiederholung von Anweisungen oder eine Verzweigung des Programmablaufs unter einer bestimmten Bedingung. Auf solche Programmstrukturen werden wir in Lektion 9 näher eingehen.

Mit Programmablaufplänen lassen sich alle denkbaren Strukturen darstellen, mit Struktogrammen hingegen nur einige bestimmte. Um die Übersichtlichkeit, Änderbarkeit und Zuverlässigkeit von Programmen zu verbessern, sollte man nicht beliebige Programmstrukturen verwenden, sondern nur bestimmte, nämlich genau die durch Struktogramme darstellbaren Strukturen. Durch die Beschränkung auf diese Strukturen befolgen wir die Regeln der **Strukturierten Programmierung**.

Die Bedeutung der verschiedenen Sinnbilder bei Struktogrammen werden Sie im Laufe der Fibel kennenlernen. Sie kann auch im Anhang A nachgeschlagen werden. Die äußere Form jeden Sinnbildes ist ein Rechteck. Die obere Kante des Rechtecks bedeutet den Beginn der Verarbeitung, die untere Kante das Ende der Verarbeitung. Das einfachste Sinnbild sieht demnach so aus:

Die Sinnbilder können ineinander verschachtelt werden. Schachtelt man beispielsweise mehrere Sinnbilder des Typs "Verarbeitung" in ein großes Sinnbild dieses Typs, so erhält man eine (Verarbeitungs-) "Folge". Als einfaches Beispiel für eine **Folge** von Verarbeitungsschritten wollen wir uns hier das Struktogramm zu unserem Beispielalgorithmus ansehen:

maximaler Abstand

drei Zahlen einlesen
die größte Zahl bestimmen
die kleinste Zahl bestimmen
den Abstand durch Differenzbildung bestimmen
den Abstand ausgeben

Eine Verfeinerung kann nun einfach vorgenommen werden, indem man ein neues Struktogramm zeichnet und es mit einem entsprechenden Titel versieht:

die größte Zahl bestimmen

die größere von den ersten beiden Zahlen bestimmen
die größere von der soeben gewählten und der dritten Zahl bestimmen

Entsprechend müßte das Struktogramm für die Bestimmung der kleinsten Zahl aussehen. Man kann die Verfeinerung natürlich auch gleich in den Grobentwurf des Algorithmus eintragen, sofern dort noch ausreichend Platz ist.

Kontrollaufgaben

K.2.7 Welche Forderung muß ein Lösungsweg für eine Problemstellung erfüllen ?

K.2.8 Wie sollte man bei der Herleitung eines Lösungsweges, der programmiert werden soll, verfahren ?

K.2.9 Was gehört neben der Herleitung des Lösungsweges noch zum Programmentwurf ?

K.2.10 Zeichnen Sie ein einfaches Struktogramm, das beschreibt, was Sie tun, wenn Sie mit Ihrem Auto fahren wollen. Es soll beginnen mit "Autotür aufschließen" und enden mit "Motor starten". Bitte vergessen Sie nicht, den Sicherheitsgurt anzulegen!

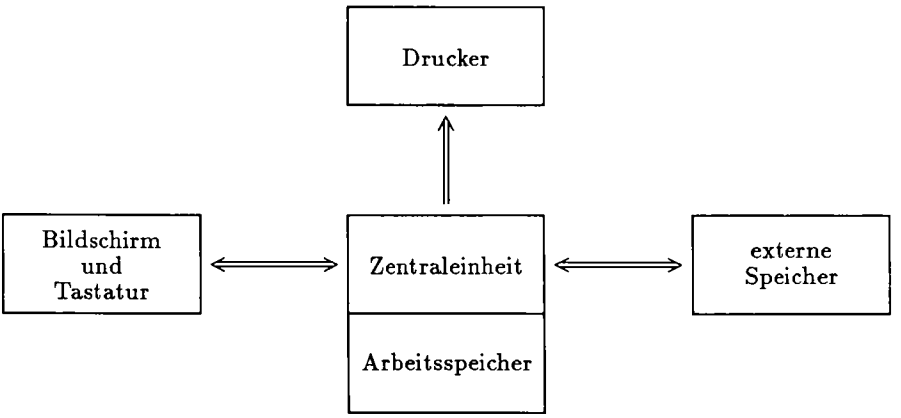
Lektion 3

Programmierumgebung

Unter einer Programmierumgebung ist im weitesten Sinne alles zu verstehen, was man benötigt, um eigene Programme entwickeln und testen zu können. Natürlich gehört dazu vor allem ein Rechner mit bestimmten Gerätekomponenten: die sogenannte **Hardware**. Dazu gehören aber auch etliche Programme, die zur **Grundsoftware** des Rechners zählen und die eine Kommunikation mit dem Rechner überhaupt erst möglich machen.

Hardware

Als wesentlich für die Programmierumgebung sind die abgebildeten Teile eines Rechners anzusehen:



Die Abbildung gibt nur den funktionalen Zusammenhang zwischen den betreffenden Komponenten wieder und stellt nicht unbedingt ihre räumliche Anordnung dar. So sind beispielsweise bei einem Arbeitsplatzrechner (engl.: Personal Computer, PC) oft Zentraleinheit mit Arbeitsspeicher und externem Speicher in einem Gehäuse der Größe eines Aktenkoffers integriert, während bei einer Großrechenanlage die entsprechenden Komponenten auch heute noch mehrere Schränke füllen.

Bildschirm, **Tastatur** und **Drucker** sind Ein-/Ausgabegeräte, über die wir mit dem Rechner kommunizieren können.

Die **Zentraleinheit** (engl.: central processing unit, CPU) ist der eigentliche Kern des Rechners. Hier werden Maschinenprogramme interpretiert und ausgeführt.

Weil der **Arbeitsspeicher** flüchtig ist, d.h., alle eingegebenen Daten, insbesondere Programmtexte, nach dem Abschalten des Rechners verloren gehen, benötigt man **externe Speicher**, die Informationen auf Dauer speichern können. Externe Speicher sind zum Beispiel Festplatten (engl.: hard disk) oder Disketten (engl.: floppy disk).

Grundsoftware

Zu den Programmen, die mit dem Rechner bereits mitgeliefert werden und die für den Betrieb des Rechners unerläßlich sind, gehört das **Betriebssystem**. Dieses Programm übernimmt "Verwaltungsaufgaben", die beim Betrieb des Rechners intern anfallen. Dazu gehört insbesondere die Koordinierung und Durchführung sämtlicher Ein-/Ausgabevorgänge.

Nach dem Einschalten des Rechners bzw. nach der Anmeldung zu einer Bildschirmsitzung, wie es bei Rechnern mit Mehrbenutzerbetrieb heißt, befinden wir uns normalerweise in einem Kommandomodus. Damit kann ebenso ein zeilenweise geführter Dialog gemeint sein wie auch eine graphische Benutzeroberfläche mit Fenstertechnik, Menüs und Maus.

Die auf dieser Ebene zur Verfügung stehenden Kommandos zur Ein-/Ausgabe, Verwaltung, Übersetzung und Ausführung von eigenen Programmen können als Programmierumgebung im engeren Sinne aufgefaßt werden.

Aus Gründen der Übersichtlichkeit sind die Daten auf den externen Speichern zu **Dateien** zusammengefaßt. Eine Datei kann zum Beispiel enthalten:

- einen Programmtext in einer höheren Programmiersprache
- einen beliebigen anderen Text
- Zahlen
- ein Maschinenprogramm

Dateien tragen einen Namen, unter dem wir sie in Kommandos ansprechen können. Das Betriebssystem führt auf dem externen Speicher ein **Dateiverzeichnis**, in das alle Dateien namentlich eingetragen werden.

Hinter allen Kommandos stecken Maschinenprogramme, die zur Ausführung gebracht werden. Herausragende Bedeutung haben zwei, nämlich der **Editor** zur Eingabe und Änderung von Programmtexten und der **Übersetzer** zum Erzeugen von Maschinenprogrammen aus Quellprogrammen.

Da sich die Namen der Kommandos und ihre Verwendung von Rechner zu Rechner sehr stark unterscheiden, können wir hier nur die allgemeine Bedeutung der Kommandos erläutern. Ziehen Sie daher parallel dazu die Unterlagen des Ihnen zur Verfügung stehenden Rechners heran.

Kommandos zur Dateiverwaltung

Die hier aufgeführten Kommandos sind die wichtigsten Kommandos zur Dateiverwaltung. Sie erleichtern Ihnen die Arbeit und helfen, den Überblick über Dateien und Programme zu bewahren.

- Ausgabe des Dateiverzeichnisses
- Ausgabe einer Datei auf den Bildschirm
- Ausgabe einer Datei auf den Drucker
- Kopieren einer Datei
- Löschen einer Datei
- Umbenennen einer Datei

Editor

Mit dem (Text-)Editor können wir einen beliebigen Text (z.B. einen Programmtext) in den Rechner eingeben und dafür sorgen, daß dieser Text in einer (Text-) Datei abgelegt wird. Der Inhalt einer vorhandenen Datei kann mit dem Editor verändert werden. Man spricht auch von der Tätigkeit des "Edierens".

Pascal-Übersetzer

Der Übersetzer erzeugt aus einem Pascal-Quellprogramm ein ablauffähiges Maschinenprogramm. Er muß dazu intern den Quellprogrammtext in mehreren Schritten verarbeiten. Auf den Vorgang des Übersetzens kommen wir in Lektion 6 zurück.

Ausführung eigener Programme

Ein mit dem Übersetzer erzeugtes Maschinenprogramm können Sie ausführen lassen. Als Kommando reicht oft der Programmname aus; bei manchen Systemen gibt es auch ein spezielles Kommando zum Starten eines eigenen Programms.

Pascal-Systeme

Normalerweise geben Sie alle Kommandos zur Dateiverwaltung, zum Edieren, zum Übersetzen und zum Ausführen eines Programms auf der Kommandoebene des Betriebssystems. Für Pascal gibt es aber auch sogenannte Pascal-Systeme, wie zum Beispiel Turbo-Pascal oder PolyPascal. Dabei handelt es sich um Programmentwicklungssysteme, in die ein Editor, ein Übersetzer, ein sogenanntes Laufzeitsystem zum Ausführen der Programme und einige Kommandos zur Dateiverwaltung integriert sind. Wenn Sie mit einem solchen System arbeiten, dann rufen Sie auf der Kommandoebene des Betriebssystems das Pascal-System auf. Weitere Kommandos geben Sie dann auf der Kommandoebene des Pascal-Systems! Mit einem bestimmten Kommando können Sie dann auf die Ebene des Betriebssystems zurückkehren.

Kontrollaufgaben

K.3.1 Besorgen Sie sich Informationen über Ihren Rechner, Ihr Betriebssystem und eventuell Ihr Pascal-System! Machen Sie sich mit den wichtigsten Kommandos zur Dateiverwaltung und dem Editor vertraut! (Mit dem Übersetzen und Ausführen von Programmen befassen wir uns erst in Lektion 6.)

K.3.2 Erzeugen Sie eine Datei mit dem Namen `hallo`, die einen kurzen Text enthält!

K.3.3 Geben Sie die Datei `hallo` auf den Drucker aus!

K.3.4 Kopieren Sie die Datei `hallo` auf eine Datei mit dem Namen `huhu`!

K.3.5 Kontrollieren Sie, ob die Dateien `hallo` und `huhu` existieren, indem Sie sich das Dateiverzeichnis ansehen!

K.3.6 Benennen Sie die Datei `hallo` um in `holla`!

K.3.7 Löschen Sie die Datei `huhu`!

Übungsaufgabe: Bitte bearbeiten Sie jetzt die Übungsaufgabe 1!

Lektion 4

Elementare Bestandteile eines Programms

In dieser Lektion werden Sie die Grundelemente kennenlernen, aus denen Pascal-Programme zusammengesetzt werden.

Sehen wir uns zunächst ein Beispiel an:

```
program stdprojahr (output);

begin
   write(365*24,' Stunden hat ein Jahr')
end.
```

Dieses Programm präsentiert uns auf dem Bildschirm die hochinteressante Information: `8760 Stunden hat ein Jahr`. Es ist zwar ein sehr kurzes Programm, aber dennoch lassen sich bereits einige grundsätzliche Eigenschaften von Pascal-Programmen erkennen.

Ein Programm besteht aus einer Folge von **Symbolen**. Die wichtigsten Pascal-Symbole sind:

- Zeichenketten,
- Zahlen,
- Spezialsymbole,
- Bezeichner.

Zeichenketten

Zeichenketten sind Folgen von Zeichen, die links und rechts von Apostrophen ' ' eingeschlossen sind. Innerhalb von Zeichenketten können alle auf einem Rechner verfügbaren Zeichen vorkommen. Sollte ein Apostroph so übermütig sein und selbst Teil einer Zeichenkette sein wollen, so muß es sich zweimal hintereinander in die Kette einreihen.

Beispiele:
```
'Dies ist eine Zeichenkette.'
'?'
'1.000.000.-- $'
'0123456789'
'That''s it!'
```

Zahlen

Bei den Zahlen müssen wir die ganzen Zahlen und die "reellen" Zahlen unterscheiden.

Die **ganzen Zahlen** werden in der üblichen Weise mit oder ohne Vorzeichen dargestellt:

```
45   +777   -28375    (auch 007 ist zulässig)
```

Der Wertebereich der ganzen Zahlen ist in Abhängigkeit vom verwendeten Rechner und Übersetzer unterschiedlich. Die üblichsten Wertebereiche sind -32768 ... +32767 und -2147483648 ... +2147483647.

Die **reellen Zahlen** können entweder in der Festpunktschreibweise

```
3.14159   0.00125   -2345678.0    +17.5
```

oder in der Gleitpunktschreibweise

```
1e-12   123.0E+9   -45.678e7    +21E6
```

dargestellt werden. `1e-12` bedeutet dann also "1 mal 10 hoch -12". Wichtig ist auch, daß vor und nach dem Dezimalpunkt (kein Komma!) mindestens eine Ziffer stehen muß, gegebenenfalls eine Null.

Der zulässige Wertebereich ist auch für die reellen Zahlen rechner- bzw. übersetzerabhängig (vgl. Lektion 11).

Spezialsymbole

Die in Pascal zulässigen Spezialsymbole sind:

```
+   -   *   /   :=   .   ,   ;   :   =   <>
<   <=  >=  >   (   )   [   ]   ^   ..
```

Ebenfalls zu den Spezialsymbolen gehören die sogenannten **Wortsymbole**. In Pascal gibt es die folgenden 35 Wortsymbole (auch **reservierte Wörter** genannt):

```
and       downto      if        or          then
array     else        in        packed      to
begin     end         label     procedure   type
case      file        mod       program     until
const     for         nil       record      var
div       function    not       repeat      while
do        goto        of        set         with
```

Neben diesen reservierten Wörtern aus Standard-Pascal sind in den verschiedenen Pascal-Versionen noch ein paar weitere Wörter reserviert. In Ihrem Handbuch finden Sie eine Liste der reservierten Wörter.

Die Bedeutung der einzelnen Spezialsymbole wird im weiteren Verlauf der Fibel bei passender Gelegenheit erläutert werden.

Bezeichner

Bezeichner werden verwendet, um bestimmten Objekten Namen zu geben. Solche Namen können Sie frei Ihrer Phantasie entspringen lassen, solange sie nur aus Buchstaben (a–z, A–Z) und Ziffern (0–9) bestehen und mit einem Buchstaben beginnen. Bezeichner dürfen beliebig lang sein.

Unser Beispielprogramm hat den Namen `stdprojahr` bekommen. Es hätte auch `spinat` heißen können, aber erstens ist Spinat nicht jedermanns Sache und zweitens sollten Namen so gewählt werden, daß sie etwas über die Bedeutung des benannten Objekts aussagen.

Natürlich dürfen Sie keine Wortsymbole als Bezeichner verwenden. Sollten Sie versehentlich doch eines dieser reservierten Wörter als Bezeichner mißbrauchen wollen, wird Sie ihr Pascal-Übersetzer nicht besonders höflich, aber direkt auf Ihren Irrtum hinweisen.

Weiterhin gibt es eine Reihe von **vordefinierten Bezeichnern** (Standardbezeichner), denen von vornherein eine bestimmte Bedeutung zugewiesen ist. Beispielsweise kann unter dem Namen `maxint` der größte darstellbare ganzzahlige Wert angesprochen werden. Im Gegensatz zu den reservierten Wörtern können Sie derartige Namen in Ihrem Programm mit einer neuen Bedeutung versehen, wobei die vordefinierte Bedeutung natürlich verlorengeht. Eine solche Umdefinition scheint nicht sonderlich sinnvoll, es sei denn, Sie möchten sich und andere bei einem späteren Lesen des Programms ein wenig in Verwirrung stürzen.

Einige Beispiele für Pascal-Bezeichner:

gültige:	`Summe`	
	`VERSION123`	
	`x1`	
	`kubikwurzel`	
ungültige:	`Std pro Jahr`	(enthält Leerzeichen)
	`Ping-Pong`	(enthält Spezialzeichen)
	`100stel`	(beginnt nicht mit Buchstaben)
	`file`	(reserviertes Wort)

Groß- und Kleinbuchstaben werden in Pascal nicht unterschieden.
`ChopSuey` und `chopsuey` sind also derselbe Name.

Trennung der elementaren Bestandteile

Schließlich ist da noch etwas, was uns an unserem Beispielprogramm hätte auffallen können. Es ist die **Zeilenstruktur** des Programms. Diese Zeilenstruktur hat auf Pascal-Programme prinzipiell keinen Einfluß. Man könnte das ganze Programm auch in eine Zeile schreiben (vorausgesetzt sie ist lang genug) oder es irgendwie anders auf mehrere Zeilen verteilen. Man sollte die Darstellung eines Programms aber immer so wählen, daß die Lesbarkeit und Übersichtlichkeit des Programms erhöht wird. Wir werden darauf später noch eingehen.