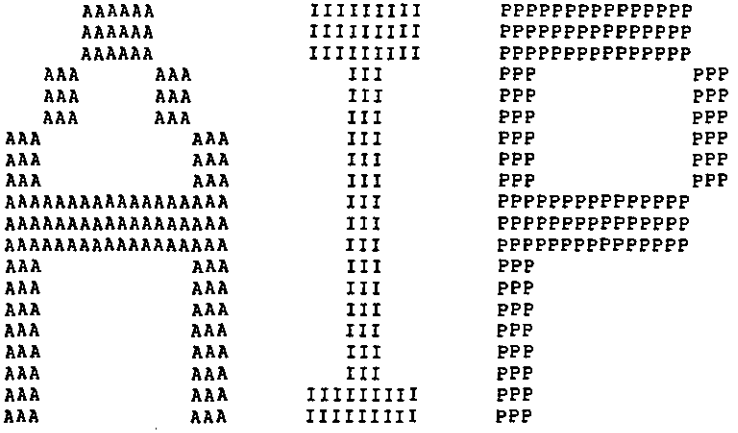
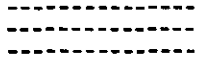
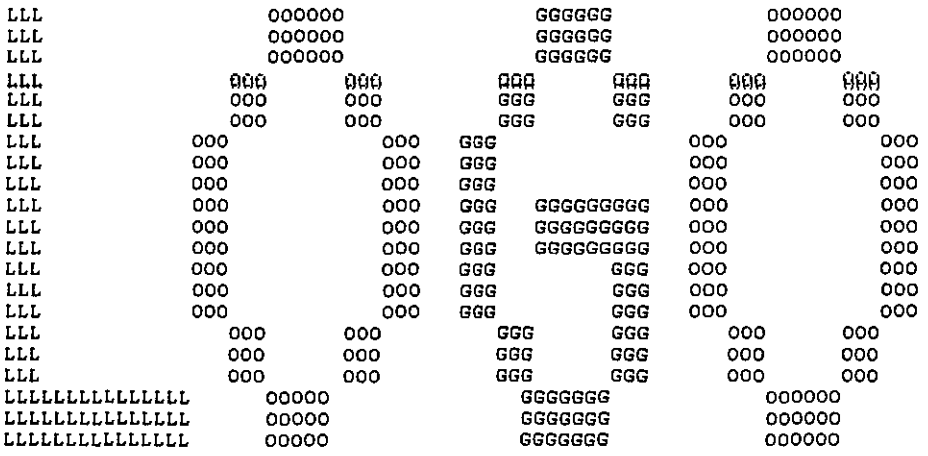






A I P - L O G O

Version für alle geläufigen Z80 Systeme
programmiert 1985.

xrr / BBG-Software

Schimmelmannstr. 90

2070 A H R E N S B U R G

I N H A L T :

V O R W O R T :

Hinter dem Namen LOGO steht die Verwirklichung der Vorstellung, eine Programmiersprache zu schaffen, die besonders Kindern und Schülern den Einstieg in die Welt der Computer erleichtert. AIP - LOGO wird auch gerade als deutsches LOGO diesen Ansprüchen besonders gerecht.
Hierbei spielt die Implementierung der "Igelgrafik" eine große Rolle. Dadurch ist es möglich, die Wirkung von Befehlen und Programmen direkt auf dem Bildschirm zu verfolgen. Dies ist schon bei den einfachsten Beispielen möglich.
Aber AIP - LOGO ist ebenso eine hochentwickelte Programmiersprache, mit der auch sehr viele Ideen der künstlichen Intelligenz voll ausgeschöpft werden können. Obwohl sich die Struktur der Prozeduren von AIP - LOGO wesentlich von z.B. LISP unterscheidet, ist AIP - LOGO der ideale Einstieg für die Entdeckung der KI (künstlichen Intelligenz). Dieses wird nur dadurch möglich, daß sich die Strukturen von Daten und Programmen intern nicht unterscheiden. Bei nichtprozeduralen Spachen wie FORTRAN oder BASIC ist so etwas nicht denkbar.

Die wichtigsten Eigenschaften von AIP - LOGO:

- AIP - LOGO ist eine interaktive prozedurale Sprache.
  Logoprogramme setzen sich aus Prozeduren zusammen, die wiederum aus Logogrundwörtern gebildet werden. Diese Prozeduren können dann wie die Grundwörter wieder in Prozeduren aufgerufen werden. Dadurch kann ein sehr komplexes Programm entstehen. Die Logoprozeduren können mit Eingaben versehen werden und können außerdem berechnete Werte oder Daten zurückgeben.
- AIP - LOGO ist eine listenverarbeitende Sprache.
  Die in AIP - LOGO existierenden Objekte beschränken sich nicht nur auf Zahlen und Wörter. Es ist vielmehr auch möglich, zusammengesetzte Strukturen, die Listen, zu bilden.
  Eine Liste ist die Aufreihung von Objekten in einer bestimmten Ordnung. Hierbei können diese Objekte Zahlen, Wörter oder auch wiederum Listen sein. Diese Listen sind mit einigen wenigen Befehlen schnell und einfach zu manipulieren. Insbesondere sind auch Prozeduren als Listen darstellbar und Listen als Prozeduren interpretierbar. Dadurch ist es möglich, eine direkte Kontrolle über die Ausführung von Befehlen und Prozeduren zu haben. Es ist auch möglich, spezielle schnittstellenähnliche Programme zu schreiben, die zur Kommunikation mit sehr jungen oder auch mit behinderten Kindern geeignet sind.

Ein weiterer wichtiger Punkt bei AIP - LOGO ist die schon erwähnte Igelgrafik.

Der Igel (im Englischen turtle = Schildkröte) ist ein kybernetisches Objekt, welches sich auf dem Bildschirm wie ein "lebendiges Tier" bewegen kann. Dabei hinterläßt er - der Igel - je nach Wunsch eine Linie. Seine Bewegungen können durch die Befehle RECHTS, LINKS, VORWAERTS, RUECKWAERTS gesteuert werden. Dieser Igel ist auf dem Bildschirm sichtbar und es ist auch die Richtung zu erkennen, in die er sich bewegen wird. Durch folgende kleine Prozedur zeichnet der Igel ein Quadrat.

```
PR QUADRAT
 WIEDERHOLE 4 [VORWAERTS 50 RECHTS 90]
ENDE
```

Diese Prozedur veranlaßt den Igel, sich viermal nacheinander erst um 50 Einheiten vorwärts zu bewegen und dann um 90 Grad nach rechts zu drehen. Das Resultat ist ein Quadrat.

Die Igelgrafik hat sich als pädagogisches Mittel sehr bewährt, um Kinder an den Computer heranzuführen und sie die ihm eigene mathematische Logik verstehen zu lassen.
Die Igelgrafik bietet auch grenzenlose Möglichkeiten, alle Techniken des Programmierens anschaulich darzustellen und zu üben. Das Üben erfolgt hier spielerisch, indem versucht wird, den Igel bestimmte Figuren zeichnen zu lassen und die dazu notwendigen Prozeduren zu definieren. Auch einige Spiele lassen sich gut auf AIP - LOGO programmieren.
Diese Beschreibung wird die Möglichkeiten, die AIP - LOGO bietet, ebenfalls mit Beispielen aus der Igelgrafik erläutern.

Zur Entstehung von AIP - LOGO:
Die deutschen Grundwörter für die Befehle in AIP - LOGO basieren im wesentlichen auf dem Standart, der an der Pädagogischen Hochschule Esslingen in einem LOGO-Projekt 1982/83 entwickelt wurde. Dieser Standart wurde als erstes in IWT-LOGO auf den Computer Apple II realisiert. Der Autor dankt dem IWT-Verlag für die freundliche Erlaubnis, diesen Standart übernehmen zu dürfen. AIP - LOGO wurde zu IWT-LOGO aufwärts kompatibel geschrieben. Schon aufgrund der Verwendung des leistungsfähigeren Z-80 Microprozessors ergaben sich weitergehende Möglichkeiten für diese Programmiersprache.

Ich hoffe, daß Sie mit dieser LOGO - Version zufrieden sein werden und auch, daß dieses Anleitungsbuch für alle verständlich genug geschrieben ist.
Für Rechtschreibfehler u.ä. möchte ich mich schon an dieser Stelle entschuldigen. - Ich schreibe in der Regel nur Programme und keine Bücher.

Und nun wünsche ich Ihnen viel Spaß mit AIP - LOGO.

xrr
der Autor.

1 Grundlagen von AIP - LOGO

In diesem ersten Kapitel sollen alle technischen Details von AIP - LOGO auf Ihrem Microcomputer beschrieben werden.

1.1 Starten von AIP - LOGO

Sie haben die Quick-disk Version für Ihren MZ-800 erworben:
Sie brauchen die Quick-disk nur einzulegen. Wenn Sie dann den Rechner einschalten, wird AIP - LOGO automatisch geladen und gestartet.

Sie haben die Cassetten Version für Ihren MZ-800 erworben:
Schalten Sie Ihren Rechner ein und drücken Sie "C" für Cassette. Dann legen Sie die AIP - LOGO Cassette ein und drücken PLAY. AIP - LOGO wird nun geladen und startet sich selbst.

AIP - LOGO meldet sich mit :

```
 ...
 Hallo, hier ist LOGO !
?
```

1.2 Erste Versuche mit AIP - LOGO

Wenn AIP - LOGO bereit ist, einen Befehl zu erhalten, wird dieses durch das Fragezeichen "?" am linken Bildschirmrand und den dahinter blinkenden "Blinker" (Cursor) gemeldet.
Man kann jetzt einen Befehl eintippen. Die Eingabezeile wird durch Drücken der Taste "CR" beendet.

ACHTUNG:
Drücken Sie die Tasten immer nur kurz. Ansonsten erhalten Sie sehr schnell mehrfach den gewünschten Buchstaben (Dauertastenabfrage).

Wir wollen ausrechnen, was die Summe von 45 und 78 ist. (45 + 78)
Dazu geben wir hinter dem Fragezeichen ein:

```
DRUCKEZEILE 45 + 78                    "CR"
```

Wir geben also folgende Tastenfolge ein:
D,R,U,C,K,E,Z,E,I,L,E,Leertaste,4,5,Leertaste,+,Leertaste,7,8,"CR".
Der Befehl DRUCKEZEILE veranlaßt AIP - LOGO, eine Zeile zu drucken. Der Inhalt dieser Zeile steht als Eingabe hinter dem Befehl DRUCKEZEILE.
AIP - LOGO rechnet das Ergebnis von 45 + 78 aus und gibt es auf den Bildschirm aus.

Es ergibt sich folgendes Bild:

```
?DRUCKEZEILE 45 + 78                   ; eingegebene Zeile
123                                    ; gedruckte Zeile von LOGO
?...                                   ; Hinter dem Fragezeichen erscheint
                                         wieder das blinkende Feld - der
                                         Blinker -, was bedeutet, daß LOGO
                                         wieder bereit ist.
```

ANMERKUNG: Ab jetzt wird in den gezeigten Beispielen alles, was Sie in den Computer eintippen müssen, in Schrägschrift geschrieben. Das, was AIP - LOGO auf den Bildschirm ausdruckt, wird - wie der Text - in normalen Zeichen geschrieben. Dies erhöht die Übersichtlichkeit dieses Handbuches.

AIP - LOGO kennt die mathemathischen Regeln Punkt vor Strich und rechnet auch erst den Inhalt von Klammern aus. Die Symbole sind "+" für "PLUS", "-" für "MINUS", "*" für "MAL", "/" für "DURCH" und "**" für "HOCH".

Ein Beispiel :

```
   2 + 3 * (5 + 1) + (2 - 4)
 = 2 + 3 * 6 + -2
 = 2 + 18 - 2
 = 18
```

Machen wir folgenden Versuch:

```
?DRUCKEZEILE 2 + 3 * (5 + 1) + (2 - 4)
18
?...
```

AIP - LOGO druckt das richtige Ergebnis aus.

Ebenso kann DRUCKEZEILE auch ein Wort oder einen Satz ausgeben. Dazu muß allerdings ein Anführungszeichen vor das Wort geschrieben werden. Ein Satz muß in eckige Klammern gefaßt werden.

```
?DRUCKEZEILE [AIP - LOGO IST EINE SPRACHE]
AIP - LOGO IST EINE SPRACHE
?DRUCKEZEILE "WORT
WORT
?...
```

Besonders wichtig sind die Leerstellen. Hätten Sie z.B. aus Versehen eine vergessen, würde folgendes erscheinen:

```
?DRUCKEZEILE45 + 78                          ; eingegebene Zeile
Prozedur DRUCKEZEILE45 nicht bekannt !       ; Fehlermeldung
?...
```

AIP - LOGO meldet den Fehler. AIP - LOGO hat gedacht, daß die 45 mit zum Namen des gewünschten Befehls gehört. Dieser Befehl war aber nicht bekannt.

1.3 LOGO und die Tastatur

Da die Benutzung der Tastatur in LOGO einige Besonderheiten birgt, ist ihr hier ein eigener Abschnitt gewidmet. Zuerst wollen wir uns ansehen, wie eine Befehlszeile richtig ewingegeben wird:

1.3.1 Eingeben und Korrigieren einer Befehlszeile

In AIP - LOGO haben Sie die volle Freiheit, mit den Pfeiltasten über den Bildschirm zu wandern. Der Blinker bewegt sich dann immer in Richtung der gedrückten Pfeiltaste.
Auch können Sie immer die Tasten "INST" und "DEL" sowie "CLR" (SHIFT+"INST") und "HOME" (SHIFT+"DEL") benutzen.

Die Taste "INST" fügt vor das Zeichen, auf dem der Blinker steht, ein Leerzeichen ein. Dabei rückt die restliche Zeile hinter dem Blinker um eine Spalte nach rechts. Der Blinker steht dann auf dem leeren Feld.

Die Taste "DEL" löscht das Zeichen vor dem Blinker. Dabei rückt der Blinker und die gesamte Zeile um ein Zeichen nach links.

Die Taste "CLR" ("SHIFT"+"INST") löscht den Bildschirm vollständig. Dabei wird der Blinker in die linke obere Ecke des Bildschirms gesetzt.

Die Taste "HOME" ("SHIFT"+"DEL") setzt den Blinker in die linke obere Ecke des Bildschirms.

Eine Befehlszeile gilt als eingegeben, wenn Sie die Taste "CR" drücken. Es wird dann von AIP - LOGO genau die Zeile übernommen, auf der der Blinker steht. Dadurch kann eine Zeile mehrfach eingegeben werden, ohne daß Sie sie immer wieder eintippen müssen. Sie müssen lediglich mit den Pfeiltasten wieder zurück auf die entsprechende Zeile fahren und erneut "CR" drücken.

LOGO-Grundwörter können im Gegensatz zu selbstdefinierten Prozeduren beliebig in großen oder kleinen Buchstaben eingegeben werden, nicht aber die selbstdefinierten Prozeduren und Namen.

Durch die Taste "ALPHA" kann zwischen Großschrift und Kleinschrift umgeschaltet werden.

Die Taste "TAB" hat die gleiche Wirkung, wie die Taste " ("SHIFT"+"2"). Da das Anführungszeichen sehr oft benötigt wird, ist dies sehr praktisch.

1.3.2 Tasten mit besonderer Bedeutung

In AIP - LOGO haben einige Buchstabentasten in Verbindung mit der Taste "CTRL" eine besondere Bedeutung. Dieses wird in diesem Handbuch noch im Einzelnen genau beschrieben. Ein Beispiel:

"CTRL"+"C" = "Ctrl-C" sprich: kontrol C !

Es besteht aber auch die Möglichkeit anstatt der "CTRL"-Taste mit einem Buchstaben eine einzelne sonst nicht genutzte Taste zu drücken.

"SHIFT"+"BREAK" = "Ctrl-C"

Anstatt "Ctrl-C" kann auf Ihrem MZ-800 auch immer "SHIFT"+"BRAEK" gedrückt werden - dies hat in jedem Fall die gleiche Wirkung!

"BREAK" = "Ctrl-S"

"CTRL-S" hält in AIP - LOGO eine Bildschirmausgabe an. Die Ausgabe wird auf einen beliebigen weiteren Tastendruck fortgesetzt. Sie können aber auch nur die "BREAK"-Taste drücken um damit z.B. ein Programmlisting anzuhalten.

"GRAPH" = "Ctrl-L"

"Ctrl-L" veranlaßt im AIP - LOGO - Editor das Abspeichern einer bearbeiteten Prozedur und Verlassen des Editors. Auf Ihrem MZ-800 können Sie anstatt "Ctrl-L" auch die GRAPH-Taste benutzen (dies ist auf Dauer angenehmer!).

" " = "Ctrl-Z"

"CTRL-Z" veranlaßt in AIP - LOGO das Unterbrechen eines Programms und Übergehen in den Unterbrechungsmodus. Auf dem MZ-800 können Sie auch die unbeschriftete Taste über der Taste "CR" benutzen.

RETURN

Im Folgenden wird die Taste "CR" nur noch kurz mit RETURN bezeichnet. Diese Taste wird ja benutzt, um eine Eingabe abzuschließen.

Die F-Tasten (1-10)

Als F-Tasten werden die fünf Tasten F1, F2, F3, F4 und F5 links oben bei Ihrer Tastatur bezeichnet. In Verbindung mit der "SHIFT"-Taste werden diese Tasten F6, F7, F8, F9 und F10 genannt.

1.4 Der IGEL

Nun wollen wir uns der schon beschriebenen Igelgrafik zuwenden. Um sie zu starten, geben wir den Befehl "BILD" ein (mit RETURN selbstverständlich!):

?BILD

Es wird in den "Bildmodus" übergegangen. Nun wird der ganze Bildschirm gelöscht. Nur ein kleines Dreieck ist in der Mitte zu sehen. Und damit wir weiterhin Befehle eingeben können, stellt AIP - LOGO die untersten fünf Zeilen für Eingaben zur Verfügung und wartet mit einem Fragezeichen auf weitere Befehle.
Um dieses kleine Dreieck in der Bildschirmmitte dreht sich jetzt alles. Es ist - wie schon erwartet - der IGEL. Die Spitze des Dreiecks stellt die Nase des Igels dar. Der Igel schaut also in die Richtung dieser Spitze und wird auf den Befehl VORWAERTS in genau diese Richtung laufen. Es ist das lebendige Wesen von LOGO.
Mit dem Befehl

?LOESCHESCHIRM
...

(ohne Eingaben) kann wieder in den "Befehlsmodus" zurückgesprungen werden. Der Igel verschwindet wieder, und es steht der ganze Bildschirm für die Eingabe von Befehlen zur Verfügung.

1.4.1 Die einfachsten Igelbefehle

Um die Bewegungen des Igels auf dem Bildschirm zu steuern, gibt es eine Vielzahl von Befehlen. Die vier einfachsten sollen jetzt erläutert werden: Mit RECHTS und LINKS kann der Igel um seine eigene Achse gedreht werden, ohne daß er seinen Standort verändert. VORWAERTS und RUECKWAERTS lassen den Igel in Richtung seiner Nase bzw. zurück laufen.

Machen Sie folgenden Versuch:

?BILD
?VORWAERTS 50
?RECHTS 45
?VORWAERTS 25
?...

Nach dem ersten Befehl hat sich der Igel um ein Stück in Richtung seiner Nase nach oben bewegt (s. Bild 1.1). Nach dem zweiten dann um 45 Grad nach rechts gedreht und dann nach dem dritten Befehl um nochmal die halbe Stecke wieder in Richtung seiner Nase nach rechts oben bewegt. Da steht er jetzt und wartet auf weitere Befehle.

Der Igel darf übrigens nicht den Bilschirmrand überschreiten. Wenn Sie dem Igel befehlen, über den Bildschirmrand hinaus zu gehen, führt dies zu der Fehlermeldung "Igel ist im Aus !" und der Befehl wird n i c h t ausgeführt.

(Bild 1.1)

nach VORWAERTS 50 nach RECHTS 45 nach VORWAERTS 25

1.4.2 Die Kontrolle des Igels

Es gibt eine Menge Befehle, um den Igel und seine Spur zu beeinflussen. An dieser Stelle sollen genannt werden:

- STIFTAB
- STIFTHOCH
- VERSTECKIGEL
- ZEIGIGEL
- VOLLBILD
- TEILBILD

- FARBE

Geben Sie STIFTHOCH ein und dann den Befehl VORWAERTS 50:

```
?STIFTHOCH
?VORWAERTS 50
?STIFTAB
?VORWAERTS 25
?...
```

Der Igel bewegt sich zwar, wie wir es erwartet haben, aber er hinterläßt keine Spur! Der Befehl STIFTHOCH veranlaßt also, daß der Igel bei seinen Bewegungen keine Spuren zeichnet. Der Befehl STIFTAB bewirkt das Gegenteil, nämlich daß der Igel eine Spur hinterläßt. Man kann sich STIFTHOCH und STIFTAB so vorstellen, daß der Igel seinen STIFT, mit dem er die Linien zeichnet, hochnimmt bzw. absetzt.

?VERSTECKIGEL
?VORWAERTS 20
?ZEIGIGEL
?...

Der Befehl VERSTECKIGEL läßt den Igel vom Bildschirm verschwinden, d.h. er verschwindet nicht wirklich, er ist lediglich nicht mehr sichtbar. So ist er nach wie vor in der Lage, bei seinen Bewegungen Linien zu zeichnen. Auf den Befehl ZEIGIGEL taucht der Igel wieder auf.

Anstelle der Befehle VERSTECKIGEL und ZEIGIGEL kann auch "Ctrl-A" gedrückt werden. Dies hat den gleichen Effekt: Der Igel ist nicht mehr zu sehen oder eben wieder zu sehen. Es wird immer zwischen ZEIGIGEL und VERSTECKIGEL hin- und hergeschaltet.
Dies ist auch während der Programmausführung möglich!

("Ctrl-A" bedeutet das gleichzeitige Drücken von der "CTRL" und der "A" Taste.)

Auch die uns schon bekannten Befehle BILD und LOESCHESCHIRM kontrollieren die Igelgrafik. Die Grafik wird durch diese Befehle ein- und ausgeschaltet.

Es gibt in AIP - LOGO zwei Grafikmodi:

den VOLLBILD-Modus und
den TEILBILD-Modus.

Geben Sie ein:

?VOLLBILD
...

Sie sehen, daß die Schrift kurz vollständig verschwindet. AIP - LOGO ist kurz in den VOLLBILD-Modus übergegangen. Im VOLLBILD-Modus steht der gesamte Bildschirm nur dem Igel und seiner Grafik zur Verfügung. Es ist keine Schrift zu sehen. Trotzdem existieren die fünf Zeilen für die Schrift noch, auch wenn sie nicht mehr sichtbar sind.

Mit dem Befehl TEILBILD kann wieder in den TEILBILD-Modus zurückgekehrt werden.

Daß AIP - LOGO in dem Beispiel oben sofort wieder in den TEILBILD-Modus übergegangen ist, liegt daran, daß AIP - LOGO, wenn eine Eingabe erwartet wird, immer automatisch in den TEILBILD-Modus zurückkehrt, damit Sie die Befehlszeile, die Sie eingeben, auch sehen können.

Es besteht auch hier die Möglichkeit mit einem Tastendruck zwischen den BILD-Modi umzuschalten. Es kann mit "Ctrl-B" (B wie BILD) immer zwischen dem VOLLBILD-Modus und dem TEILBILD-Modus hin- und hergeschaltet werden. Auch "Ctrl-B" kann auch während der Programmausführung benuzt werden. Sie können mit "Ctrl-B" auch während Sie eine Befehlszeile eingeben, in den VOLLBILD-Modus - und zurück - schalten.

Mit dem Befehl

FARBE

kann die Farbe des "Stiftes" verändert werden. Der Befehl braucht eine ganze Zahl als Eingabe.

Diese Zahl entspricht dann einer Farbe (s. Tabelle 1.1).

Man kann sich das etwa so vorstellen: Der Igel hat mehrere Farbstifte zur Verfügung (Anzahl je nach Modus / Grafikerweiterung), die er frei wählen kann. Mit dem Befehl FARBE wählen wir nun einen dieser Farbstifte aus.

(Tab. 1.1)

	40 Zeichen	80 Zeichen	40 Z. (m.Erw.)	80 Z. (m.Erw.)
0	schwarz	schwarz	schwarz	schwarz
1	rot	helles weiß	blau	rot
2	grün	*	rot	grün
3	helles weiß	*	rosa	helles weiß
4	*	*	grün	*
5	*	*	türkis	*
6	*	*	gelb	*
7	*	*	weiß	*
8	*	*	grau	*
9	*	*	helles blau	*
10	*	*	helles rot	*
11	*	*	helles rosa	*
12	*	*	helles grün	*
13	*	*	helles türkis	*
14	*	*	helles gelb	*
15	*	*	helles weiß	*

* = nicht zugelassener Wert
m.Erw. = mit der Grafikerweiterung

Ein Beispiel:

```
?FARBE 1
?VORWAERTS 20
?...
```

Es wird eine rote (40-Zeichenmodus / ohne Erweiterung!) Linie gezeichnet.

1.5 Fehlermeldungen von AIP - LOGO

AIP - LOGO hat ein sehr komfortables Fehlermeldesystem:
Wenn AIP - LOGO nicht in der Lage ist, eine Befehlszeile zu verstehen bzw. auszuführen, wird dieses gemeldet. Die auftretenden Fehler werden von AIP - LOGO sehr ausführlich beschrieben.

Einige Beispiele:

```
?DRUCKEZEILE 7 +
+ braucht Zahl dahinter !
?...
```

Da AIP - LOGO vor und hinter jedem mathematischen Symbol (+,-,*,/,**) eine Zahl erwartet, meldet es einen Fehler, wenn keine da ist. Ebenso, wenn anstatt einer Zahl ein Wort oder eine Liste dahinter steht.

```
?LACHE
 Prozedur LACHE nicht bekannt !
?...
```

Da AIP - LOGO "LACHE" für einen Befehl hält, sucht es diesen unter den bekannten Wörtern. Da LACHE kein LOGO-Grundwort ist und auch nicht als Prozedur definiert wurde, wird es als "nicht bekannt" gemeldet.

```
?REECHTS 90
 Prozedur REECHTS nicht bekannt !
?...
```

Wie auch LACHE für AIP - LOGO nicht bekannt ist, so wird auch für REECHTS ein Fehler gemeldet. Hierbei handelt es sich aber aller Wahrscheinlichkeit nach nur um einen Tippfehler. AIP - LOGO ist nicht in der Lage, "echte" Fehler von Tippfehlern zu unterscheiden und meldet somit alle.

```
?DRUCKEZEILE 4 / 0
 Ergebnis zu gross/klein von : /
?...
```

Hier wurde eine nicht erlaubte mathematische Operation, die Division durch null, durchgeführt. Alle Fehler, die bei mathematischen Funktionen und Operationen auftreten, werden durch eine solche Fehlermeldung mitgeteilt.

Bei sehr komplexen Programmen sind die Fehler meistens nicht so einfach zu finden. AIP - LOGO bietet hierbei aber viele Hilfen, die es dann erheblich erleichtern, dem Fehler auf die Spur zu kommen. Diesem Thema werden wir uns im Kapitel 8.2 noch sehr ausführlich widmen.

1.6 Abkürzungen

Sie werden sicherlich schon gemerkt haben, wie lästig es ist, die teilweise langen AIP - LOGO Grundwörter wie DRUCKEZEILE oder VORWAERTS einzutippen. Deshalb gibt es für die meisten Grundwörter Abkürzungen von zwei oder drei Buchstaben. So ist die Abkürzung für DRUCKEZEILE z.B. "DZ". Es ist unbedeutend, ob Sie

```
?DRUCKEZEILE 5 * 2
10
?...
```

oder

```
?DZ 5 * 2
10
?...
```

eintippen. Die Benutzung der ausgeschriebenen LOGO-Grundwörter benötigt aber nicht (!) mehr Speicherplatz!
Hier die Abkürzungen aller bisher genannten Befehle:

DRUCKEZEILE	DZ	
BILD	BI	
LOESCHESCHIRM	LS	
VORWAERTS	VW	
RUECKWAERTS	RW	
RECHTS	RE	
LINKS	LI	
STIFTHOCH	SH	
STIFTAB	SA	
VERSTECKIGEL	VI	od. "Ctrl-A"
ZEIGIGEL	ZI	od. "Ctrl-A"
VOLLBILD	VB	od. "Ctrl-B"
TEILBILD	TB	od. "Ctrl-B"
WIEDERHOLE	WH	

1.7 Der Befehl WIEDERHOLE

Der WIEDERHOLE-Befehl ist einer der wirkungsvollsten von LOGO.

```
?WIEDERHOLE 3 [DZ 3 / 2]
1.5
1.5
1.5
?...
```

Die Zahl hinter WIEDERHOLE bestimmt, wie häufig der dann folgende Befehl ausgeführt werden soll. Der folgende Befehl muß wie ein Satz in eckige Klammern geschrieben werden. In diesem kleinen Beispiel sollte das Ergebnis aus 3 / 2 dreimal ausgedruckt werden. Hierbei wird das Ergebnis jedesmal neu berechnet.

1.8 Zeilen mit mehr als einem Befehl

Es ist völlig problemlos, mehrere Befehle in eine Zeile zu schreiben, es verringert nur die Übersichtlichkeit der Zeile. Wichtig ist jedoch, daß die einzelnen Befehle durch Leerstellen voneinander getrennt sind.

```
?BILD
?VORWAERTS 30 RECHTS 90 VORWAERTS 30
?...
```

Hier werden alle drei Befehle von AIP - LOGO nacheinander von links nach rechts abgearbeitet. Trifft AIP - LOGO dabei auf einen Fehler, so wird die Ausführung an der Stelle abgebrochen, wo der Fehler aufgetreten ist. Alle folgenden Befehle werden ignoriert.

1.9 Befehle in Klammern

Wenn wir den Befehl DRUCKEZEILE in runden Klammern aufrufen, so können wir eine beliebige Anzahl von Eingaben mit in die Klammern setzen. Durch die runden Klammern weiß AIP - LOGO, daß alles, was innerhalb der Klammern steht, mit zum Aufruf des Befehls DRUCKEZEILE gehört.
Zwei Beispiele:

```
?(DZ 45 + 78 [DIES IST EINE LISTE])          ; Beispiel 1
123 DIES IST EINE LISTE
?(DZ 22 "WORT 3 [EIN SATZ] 0 0 0)            ; Beispiel 2
22 WORT 3 EIN SATZ 0 0 0
?...
```

Im Beispiel 1 wurden zwei Eingaben für den Befehl DRUCKEZEILE gegeben. Die erste Eingabe (45 + 78) wurde von AIP - LOGO erst ausgerechnet und dann ausgegeben. Die zweite wurde - mit einer Leerstelle von der ersten getrennt - noch in die gleiche Zeile geschrieben. Alle Eingaben für den Befehl DRUCKEZEILE werden in die gleiche Zeile geschrieben. Im zweiten Beispiel werden sogar sieben Eingaben in die gleiche Zeile geschrieben. Es könnte der Befehl DRUCKEZEILE auch mit keiner Eingabe in den runden Klammern aufgerufen werden. Es wird dann nur ein Zeilenvorschub gegeben, ohne daß in der Zeile etwas gedruckt wird.

```
?(DRUCKEZEILE)
                                             ; leere Zeile
?...
```

VERALLGEMEINERUNG :
Wenn ein Befehl innerhalb von runden Klammern aufgerufen wird, so versucht AIP - LOGO (soweit dies möglich und sinnvoll ist) alle Eingaben innerhalb der runden Klammern für diesen Befehl auszuwerten.
So ist es in AIP - LOGO möglich, eine Vielzahl von Befehlen mit variabler Eingabenzahl (0,1,2,3,4... Eingaben) ausführen zu lassen, indem diese mit all ihren Eingaben in runde Klammern gefaßt werden.
Auf diese Möglichkeit wird aber bei jedem Befehl noch kurz hingewiesen.

1.10 Abbrechen von Programmausführungen

Die Ausführung eines Befehls oder einer Prozedur kann jederzeit abgebrochen werden. Dazu gibt man "Ctrl-C" ein.
AIP - LOGO meldet dann:

```
 ...
 Ausgestiegen !
?...
```

Wenn z.B. eingegeben wurde:

```
?WIEDERHOLE 10000 [DRUCKEZEILE "LOGO]
LOGO
LOGO
 ...
```

dann wird es sehr lange dauern, bis LOGO zehntausendmal den Befehl DRUCKEZEILE "LOGO ausgeführt hat. Sie können diese Ausführung jetzt mit "Ctrl-C" abbrechen und damit aus dem Programm aussteigen.

Es ist auch möglich, eine Programmausführung anzuhalten, ohne gleich das Programm abzubrechen:

- "Ctrl-S"

Drücken Sie einfach, während das Programm läuft, "Ctrl-S". Das Programm wird angehalten, und der Blinker blinkt. Wenn Sie jetzt eine andere Taste drücken, wird das Programm ohne Veränderung fortgesetzt. (Wenn Sie allerdings jetzt "Ctrl-C" drücken, wird das Programm ganz abgebrochen.) Mit "Ctrl-S" können Sie also auch umfangreichere Auslistungen anhalten, ohne sie gleich abzubrechen!

Es gibt noch eine dritte Möglichkeit, das Programm zu unterbrechen:

- "Ctrl-Z"

Durch "Ctrl-Z" wird die Ausführung eines Programms bzw. einer Befehlszeile angehalten und in den Unterbrechungsmodus gesprungen. Von dort kann u.a. mit "Ctrl-C" wieder in den Befehlsmodus zurückgesprungen werden.
Wir werden den Unterbrechungsmodus im Kapitel 8 noch genau besprechen.

2 Prozeduren

Alle Prozeduren in LOGO haben den gleichen Aufbau:

```
PR RECHTECK
 VORWAERTS 50
 RECHTS 90
 VORWAERTS 25
 RECHTS 90
 VORWAERTS 50
 RECHTS 90
 VORWAERTS 25
 RECHTS 90
ENDE
```

Jede Prozedur besteht aus einer Titelzeile (PR RECHTECK) und einem Körper. In der Titelzeile steht "PR" (für PRozedur oder PRogramm) und der ihr gegebene Prozedurname. Dieser Name muß immer verschieden von den LOGO-Grundwörtern und den schon definierten Prozeduren sein, da sonst Verwechslungen auftreten können. Sollten Sie aus Versehen ein LOGO-Grundwort als Prozedurnamen verwenden, so meldet AIP - LOGO den Fehler:

```
?LERNE RECHTS
 RECHTS ist LOGO-Grundwort !
 Unsinnige Titelzeile !
?...
```

Der Körper der Prozedur besteht aus beliebig vielen Zeilen, die jeweils auch mehrere Befehle beinhalten können. Die Prozedur RECHTECK hat acht Zeilen, in denen jeweils nur ein Befehl steht.
Das "ENDE" gehört nicht richtig mit zur Prozedur. Es dient nur der besseren Übersicht, zeigt also das Ende der Prozedur an.

2.1 Schaffen (Definieren) einer Prozedur

Um eine Prozedur neu zu erschaffen (definieren), benötigen wir den Befehl LERNE. LERNE braucht als Eingabe den Namen der neuen Prozedur. In unserem Beispiel also RECHTECK.

```
?LERNE RECHTECK
1:    ...
```

AIP - LOGO geht jetzt in den LERNE-Modus (AIP - LOGO Editor) über. In der nächsten Zeile erscheint eine "1:" und ein Stück dahinter wieder der Blinker. Im Gegensatz zum Befehlsmodus wird im LERNE-Modus kein Fragezeichen ausgedruckt, welches mitteilt, daß LOGO auf eine Eingabe wartet. Die "1:" soll anzeigen, daß jetzt die erste Zeile der (neuen) Prozedur eingegeben werden kann.

Der LERNE-Modus hat noch einige Besonderheiten. Diese werden weiter unten im Kapitel 2.3 beschrieben.

Geben Sie nun die erste Zeile ein (mit RETURN am Ende).

```
PR RECHTECK
1:    VORWAERTS 50                    ;"RETURN"
2:    ...
```

Nachdem Sie die Zeile mit "RETURN" abgeschlossen haben, erscheint in der nächsten Zeile eine "2:", und AIP - LOGO wartet auf die zweite Zeile. Die Zeilen der Prozedur sind hier also genau nummeriert. Wenn Sie den LERNE-Modus (Editor) wieder verlassen, verschwindet diese Nummerierung wieder. Die Zeilen sind nur im LERNE-Modus nummeriert, damit es einfacher ist, beim Schreiben einer Prozedur die Übersicht über die Zeilen zu behalten (siehe Editor / Kap. 2.3). Sie können jetzt die zweite Zeile eingeben.

```
 ...
2:    RECHTS 90
3:    ...
```

AIP - LOGO fragt jetzt nach der dritten Zeile. So können die restlichen sechs Zeilen eingeben werden.

```
 ...
3:    VORWAERTS 25
4:    RECHTS 90
5:    VORWAERTS 50
6:    RECHTS 90
7:    VORWAERTS 25
8:    RECHTS 90
9:    ...
```

AIP - LOGO fragt jetzt nach der neunten Zeile. Wir können jetzt noch beliebig viele Zeilen eingeben. Die Prozedur RECHTECK soll aber nur acht Zeilen haben. Wir haben also alle Zeilen eingegeben, und wollen das LERNEN jetzt beenden. Dazu muß "Ctrl-L" gedrückt werden. Drücken Sie "Ctrl-L"!

```
 ...
9:                                    ; "Ctrl-L"

Prozedur RECHTECK gelernt !
?...
```

AIP - LOGO hat die Prozedur RECHTECK jetzt gelernt und ist wieder in den Befehlsmodus gesprungen (am Fragezeichen zu sehen).
Wir können jetzt die Prozedur RECHTECK - wie jeden LOGO-Grundbefehl - aufrufen, und die Prozedur RECHTECK wird dann ausgeführt. Vorher müssen wir aber erst noch in den Bildmodus übergehen.

```
?BILD
 ...
?RECHTECK
?...
```

Auf den Befehl RECHTECK hin wird jetzt das Rechteck auf dem Bildschirm gezeichnet. Wir können jetzt auch noch ein zweites Rechteck - um 90 Grad gedreht - darüberzeichnen:

```
?RECHTS 90
?RECHTECK
?...
```

Nun wird das Rechteck - um 90 Grad gedreht - noch einmal gezeichnet. Hier wird deutlich, daß wir uns gar nicht mehr um den inneren Aufbau der Prozedur RECHTECK zu kümmern brauchen. Wir können diese Prozedur (fast) wie ein LOGO-Grundwort benutzen.

Der Befehl "PR" hat die gleiche Wirkung wie der Befehl "LERNE". Es wird in den LERNE-Modus übergegangen, und die Prozedur, dessen Name hinter PR steht, kann geschrieben werden.

2.2 Schachteln von Prozeduren

Prozeduren können beliebig geschachtelt werden. Die Prozedur RECHTECK zeichnet ein Rechteck. Wenn wir nun vier Rechtecke nebeneinander haben wollen, so können wir das mit folgender Prozedur RECHTECKE erreichen. Bevor Sie diese Prozedur ausprobieren geben Sie bitte LOESCHESCHIRM und wieder BILD ein.

```
PR RECHTECKE
 WH 4 [RECHTECK STIFTHOCH RE 90 VW 30 LI 90 STIFTAB]
ENDE
```

Wenn eine Befehlszeile zu lang wird (wie z.B. hier!) können Sie diese ruhig über den Bildschirmrand hinaus schreiben. AIP - LOGO fängt dann wieder in der nächsten Zeile an, weiß aber, daß die beiden Zeilen zusammengehören.

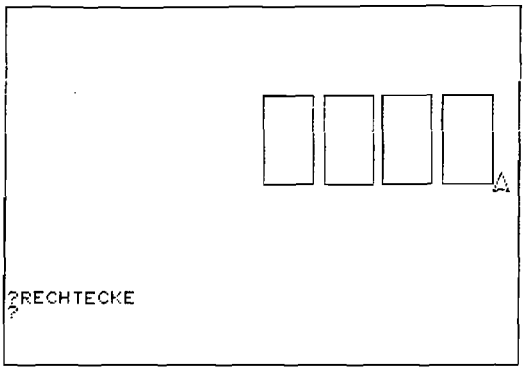
Versuchen wir jetzt, uns die Prozedur RECHTECKE einmal klar zu machen: Es soll also fogender Ablauf viermal wiederholt werden:

1. zeichne ein Rechteck (RECHTECK)
2. hinterlasse ab jetzt keine Linien mehr (STIFTHOCH)
3. drehe Dich nach rechts um 90 Grad (RE 90)
4. gehe vorwärts um 30 Einheiten (VW 30)
5. drehe Dich wieder um 90 Grad nach links zurück (LI 90).

und 6. zeichne ab jetzt wieder Linien (STIFTAB)

Die Schritte 2 bis 5 könnte man als Bewegung nach rechts um 30 Einheiten (ohne eine Linie zu zeichnen) zusammenfassen. Es werden also vier Rechtecke, um jeweils 30 Einheiten versetzt, nebeneinander gezeichnet.
Nach Aufruf der Prozedur RECHTECKE entsteht das Bild (2.1).

Die Prozedur RECHTECK wird hier in der Prozedur RECHTECKE wie ein LOGO-Grundbefehl verwendet. So können sich alle Prozeduren in LOGO gegenseitig aufrufen, so daß man tlw. sehr tief verschachtelte Programme erhält.



(Bild 2.1)

ANMERKUNG: Es können auch Kommentare in Prozedurzeilen geschrieben werden. Damit AIP - LOGO diese von Befehlen oder Listen unterscheiden kann, müssen die Kommentare in geschweifte Klammern geschrieben werden. Anstatt der ersten geschweiften Klammer kann auch ein Semikolon eingegeben werden. Dazu zwei Beispiele:

```
4:    VORWAERTS 100 {KOMMENTAR}
5:    RECHTS 90 ;DIES IST DER KOMMENTAR
```

2.3 Der Editor oder das Verändern bzw. Berichtigen von Prozeduren

Um Prozeduren zu verändern oder Fehler in Prozeduren zu berichtigen, muß man wieder in den LERNE-Modus (ab jetzt nur noch "Editor" genannt) gehen. Folgende fehlerhafte Prozedur RECHTECK soll berichtigt werden:

```
PR RECHTECK
 VORWAERTS 50
 REECHTS 90                    ; Tippfehler "E" zuviel
 VORWAERYS 25                  ; Tippfehler "Y" anstatt "T"
 RECHTS 90
 VORWAERTS 50
 RECHTS 90
 RECHTS 90                     ; eine Zeile fehlt (VORWAERTS 25)
ENDE
```

Die drei Fehler dieser Prozedur RECHTECK sind nebenstehend aufgeführt. Wir wollen dies jetzt korrigieren! Der Versuch, mit

```
?LERNE RECHTECK
 Prozedur RECHTECK ist schon bekannt!
 Unsinnige Titelzeile !
?...
```

wieder in den AIP - LOGO Editor zu kommen, mißlingt. LOGO meldet uns, daß die Prozedur RECHTECK schon bekannt ist. Wir müssen hier mit dem Befehl EDIT (Abkürzung ED) in den Editor übergehen.

?EDIT RECHTECK

```
 PR RECHTECK
1:     VORWAERTS 50
2:     REECHTS 90
3:     VORWAERYS 25
4:     RECHTS 90
5:     VORWAERTS 50
6:     RECHTS 90
7:     RECHTS 90

1:     VORWAERTS 50
```

LOGO zeigt uns die ganze Prozedur RECHTECK mit den Zeilennummern davor. Darunter wird noch einmal die erste Zeile angezeigt. Der Blinker steht auf dem "V" von VORWAERTS, und LOGO wartet auf eine Eingabe. Sie können nun mit den Pfeiltasten (nach links und rechts), aber auch mit INST und DEL, diese Zeile bearbeiten. Verändern Sie aber nicht die Zeilennummer und schreiben Sie auch nicht in den Bereich zwischen Nummer und Zeile. Wenn die Zeile nun berichtigt ist, kann mit RETURN oder "Pfeil runter" in die zweite Zeile gesprungen werden. Auch wenn nichts geändert werden soll, muß RETURN oder "Pfeil runter" eingegeben werden, um in die nächste Zeile zu springen.
Da wir erst die zweite Zeile berichtigen wollen, geben wir also erst RETURN oder "Pfeil runter" ein und berichtigen dann die zweite Zeile.

```
2:     REECHTS 90
```

Durch dreimaliges Drücken von der Taste "Pfeil rechts" gelangen wir auf das "C" von REECHTS. Nun können wir mit Hilfe von DEL den davorliegenden Buchstaben (das zweite "E") löschen. Damit ist die zweite Zeile schon berichtigt. Jetzt geben wir wieder RETURN oder "Pfeil runter" ein und berichtigen noch die dritte Zeile:

```
3:     VORWAERYS 100
```

Durch 7 mal "Rechtspfeil" kommen wir auf das fehlerhafte "Y". Wir brauchen jetzt nur noch das richtige "T" darüberzutippen.

```
3:     VORWAERTS 50                ; berichtigen, RETURN
4:     RECHTS 90
```

Die Taste RETURN oder "Pfeil runter" veranlaßt also das Springen in die nächste Zeile. Wir können aber auch in die vorherige Zeile springen. Dazu müssen wir

- "Pfeil hoch"

eintippen. Wenn wir auf der vierten Zeile stehen und dann "Pfeil hoch" eintippen, so können wir jetzt die dritte Zeile verändern. Wenn wir vor die erste Zeile springen, können wir auch die Titelzeile (PR RECHTECK) verändern.

Nun aber zu unserem zweiten Problem! Wir müssen zwischen die sechste und siebte Zeile eine weitere Zeile (VORWAERTS 25) einfügen. Dazu gehen wir mit Hilfe von RETURN und/oder "Pfeil hoch" auf die siebte Zeile. Jetzt geben wir

- "Ctrl-I" (I = Insert, engl.: einfügen)

ein, und eine neue leere Zeile sieben wird eröffnet.

```
4:    RECHTS 90                    ; RETURN
5:    VORWAERTS 50                 ; RETURN
6:    RECHTS 90                    ; RETURN
7:    RECHTS 90                    ; "Ctrl-I"
7:    ...
```

In diese neue siebte Zeile können wir nun VORWAERTS 25 eingeben. Nach RETURN oder "Pfeil runter" springt AIP - LOGO dann in die achte (ehemals siebte) Zeile. Bei dem Befehl "Ctrl-I" werden alle folgenden Zeilen um einen Platz nach hinten gerückt.

```
...
7:    VORWAERTS 50                 ; RETURN
8:    RECHTS 90                    ; "Ctrl-L"

Prozedur RECHTECK gelernt!
?...
```

Wie schon bekannt, können wir, wenn alles berichtigt ist, mit

- "Ctrl-L"

den Editor wieder verlassen. AIP - LOGO meldet, daß es die Prozedur RECHTECK nun (neu) gelernt hat.

Im Editor besteht auch die Möglichkeit, Zeilen zu löschen. Dies geschieht durch

- "Ctrl-D" ; (D = Delete, engl.: löschen)

Es wird dann die Zeile gelöscht, auf der der Blinker gerade steht. Alle hinteren Zeilen rücken dann um einen Platz nach vorne.

- "Ctrl-Z" ; (Z = zeigen)

veranlaßt keine Veränderung, es wird aber die ganze Prozedur mit Zeilennummern auf dem Bildschirm ausgegeben.

- "Ctrl-A" ; (A = Anfang)

veranlaßt das Springen auf die erste Zeile der Prozedur.

- "Ctrl-E" ; (E = Ende)

veranlaßt das Springen auf die letzte Zeile der Prozedur.

- "Ctrl-T" ; (T = Titelzeile)

veranlaßt das Springen auf die Titelzeile der Prozedur.

- "Ctrl-C"

bricht den Editor ab. Angenommen, Sie haben es sich beim Berichtigen einer Prozedur anders überlegt, und wollen die Prozedur lieber in ihrer alten Form belassen, so können Sie mit "Ctrl-C" aus dem Editor aussteigen.
AIP - LOGO fragt dann, ob wirklich abgebrochen werden soll:

```
Ausstieg ? (J/N)
...
```

Geben Sie jetzt ein "J" ein, so wird zurück in den Befehlsmodus gesprungen, und die Prozedur ist nicht (!) verändert. Bei "N" bleibt LOGO im Editor und ignoriert somit den "Ctrl-C" Befehl.

- "Ctrl-N" ; (N = neu beginnen)

startet den Editor neu. Angenommen, Sie meinen, daß Sie die Prozedur durch Ihre Änderungen nur noch verschlechtert haben, so können sie die "alte" Prozedur erneut in den Editor zurückholen. Sie können praktisch mit der Berichtigung neu beginnen. Dazu ist der Befehl "Ctrl-N" zu verwenden.

Diese ganzen Befehle können natürlich genauso benutzt werden, wenn eine Prozedur neu geschrieben wird. (Befehl LERNE ...)

2.3.1 EDIT ohne Angabe des Prozedurnamens

Wenn der Befehl EDIT ohne nachfolgende Angabe eines Prozedurnamens aufgerufen wird, so wird die Prozedur angenommen, die zuletzt im Editor bearbeitet wurde. Dabei wird auch, wenn möglich, gleich in die Zeile der Prozedur gesprungen, die zuletzt bearbeitet wurde. Dies gilt selbstverständlich ebenso bei der Abkürzung von EDIT (ED).

Mit dem Befehl

- EDF

(ohne eine Eingabe) kann an die Stelle einer Prozedur in den Editor gesprungen werden, an der zuletzt ein Fehler gemeldet wurde. Diese Möglichkeit wird aber erst weiter unten im Kapitel 8.3.1 genauer beschrieben.

Und noch eine Anmerkung:
Wenn Sie in der Titelzeile den Prozedurnamen verändern, so kann es vorkommen, daß Sie dabei einen nicht zugelassenen Namen für die Prozedur eingeben. Bei dem Versuch, den Editor dann mittels "Ctrl-L" zu verlassen, wird von AIP - LOGO sofort der Fehler

- unsinnige Titelzeile !

gemeldet, und das Verlassen des Editors wird verweigert. Sie müssen den Fehler dann erst berichtigen, oder mit "Ctrl-C" den Editor verlassen.

Eine Zusammenfassung über alle Befehle des Editors finden Sie im Anhang dieses Handbuches (Teil III.).

2.4 Fehler innerhalb von Prozedurnen

Wir haben bisher nur Fehler betrachtet, die beim direkten Eingeben von LOGO gefunden werden. In Prozeduren tauchen leider auch viel zu oft Fehler auf. LOGO meldet dann nicht nur die Art des Fehlers, sondern auch noch die Zeile, in der der Fehler aufgetreten ist, und die Prozedur, zu der die Zeile gehört, und auch noch die Schachtelungstiefe, in der der Fehler aufgetreten ist.
Diese Schachtelungstiefe erhöht sich jeweils beim Aufruf einer Prozedur um eins, und verringert sich um eins, wenn die Prozedur fertig abgearbeitet ist, oder vorzeitig aus der Prozedur ausgestiegen wurde.

Hier ein Beispiel eines Fehlers in der Prozedur RECHTECK:

```
PR RECHTECK
 VORWAERYS 50                    ; Tippfehler ("Y" anstatt "T")
 RECHTS 90
 VORWAERTS 25
 RECHTS 90
 VORWAERTS 50
 RECHTS 90
 VORWAERTS 25
 RECHTS 90
ENDE
 ...
?BILD
?RECHTECK
Prozedur VORWAERYS nicht bekannt !
  In Zeile : VORWAERYS 50
 In Aufrufebene 1 von: RECHTECK
E1?...
```

AIP - LOGO geht nun in den Fehler- (Error-) Modus über. Dieser wird im Kapitel 8 über das Fehlermeldesystem noch genauer beschrieben. Zunächst reicht für uns zu wissen, daß wir mit "Ctrl-C" zurück in den Befehlsmodus gelangen können.

Diese umfangreichen Fehlermeldungen sind sehr praktisch zum Auffinden von Fehlern in Prozeduren.
Eigentlich könnte AIP - LOGO die Prozeduren ja schon bei ihrer Definition auf Fehler hin untersuchen. Doch dies ist sehr unpraktisch, da in der Prozedur z.B. andere Prozeduren aufgerufen werden können, die erst noch geschaffen werden sollen. Daher treten die Fehler in den Prozeduren erst bei der Abarbeitung in Erscheinung.

2.5 Prozeduren mit Eingaben

```
PR QUADRAT
 WIEDERHOLE 4 [VORWAERTS 50 RECHTS 90]
ENDE
```

Die Prozedur QUADRAT zeichnet immer ein Quadrat mit der gleichen Seitenlänge 50. Stellen wir uns vor, wir wollen eine Prozedur QUADRAT1 schreiben, die nicht immer das gleiche Quadrat zeichnet sondern verschieden große. Dies ist ja für LOGO-Grundbefehle möglich: Man kann z.B. den Befehl VORWAERTS mit verschiedenen Eingaben versehen, so daß verschieden lange Linien gezeichnet werden. Wir wollen nun erreichen, daß wir auch bei jedem Aufruf der Prozedur QUADRAT1 die Seitenlänge des Quadrats miteingeben können. Dazu müssen wir die Prozedur QUADRAT1 mit einer Eingabe (der Seitenlänge) festlegen.

Grundsätzlich kann jede LOGO-Prozedur unbegrenzt viele (bis zu 250) Eingaben bekommen. Dies muß aber schon bei der Definition bestimmt werden. Die Information, daß eine LOGO-Prozedur Eingaben verlangt, wird in der Titelzeile (PR QUADRAT1 ...) mitgeteilt. Es wird hierzu jeder Eingabe ein Name zugeordnet, der mit einem großen Buchstaben beginnen muß, vor welchem (ohne eine Leerstelle!) ein Doppelpunkt stehen muß. Diese Namen werden dann hinter den Prozedurnamen in der Titelzeile geschrieben. Hierbei werden die Eingaben bei Aufruf der Prozedur in genau der Reihenfolge besetzt, wie sie hier in der Titelzeile stehen. Für die Prozedur QUADRAT1 müßte die Titelzeile folgendermaßen lauten:

```
PR QUADRAT1 :SEITENLAENGE          (Name ist beliebig!)
```

Die Prozedur QUADRAT1 benötigt hier nun eine Eingabe, die bei Aufruf unter dem Namen :SEITENLAENGE abgelegt wird und auch unter diesem Namen abgerufen werden kann. Um dies in die Prozedur zu schreiben, müssen wir im Editor mit "Ctrl-T" auf die Titelzeile springen und dann den Namen :SEITENLAENGE hinter den Prozedurnamen QUADRAT1 schreiben.

Unsere Prozedur QUADRAT1 sieht jetzt folgendermaßen aus:

```
PR QUADRAT1 :SEITENLAENGE
 WIEDERHOLE 4 [VORWAERTS 50 RECHTS 90]
ENDE
```

Wir müssen jetzt noch erreichen, daß der für die Seitenlänge wichtige Befehl "VORWAERTS 50" nicht mit 50 sondern mit dem Wert der Eingabe :SEITENLAENGE aufgerufen wird. Dies wird ganz einfach erreicht, indem statt der "50" hinter "VORWAERTS" der Name :SEITENLAENGE eingegeben wird. Die Pozedur QUADRAT1 sieht also richtig folgendermaßen aus:

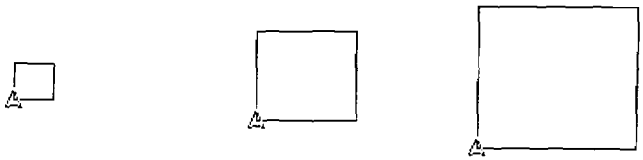
```
PR QUADRAT1 :SEITENLAENGE
 WIEDERHOLE 4 [VORWAERTS :SEITENLAENGE RECHTS 90]
ENDE
```

Nun muß bei jedem Aufruf der Prozedur QUADRAT1 ein Wert als Eingabe mitgegeben werden. Wenn Sie dies vergessen und nur

```
?QUADRAT1
Zuwenig Eingabe   für : QUADRAT1
?...
```

eingeben, dann meldet LOGO sofort den erkannten Fehler. Wenn Sie aber richtig einen Wert hinter QUADRAT1 eingeben, so wird das Quadrat immer mit der eingegebenen Seitenlänge gezeichnet (Bild 2.2).

(Bild 2.2)



QUADRAT1 20 QUADRAT1 50 QUADRAT1 80

Genauso können wir die Prozedur RECHTECK1 mit zwei Eingaben schreiben:

```
PR RECHTECK1 :LAENGE :BREITE
 VORWAERTS :LAENGE
 RECHTS 90
 VORWAERTS :BREITE
 RECHTS 90
 VORWAERTS :LAENGE
 RECHTS 90
 VORWAERTS :BREITE
 RECHTS 90
ENDE
```

Diese Prozedur RECHTECK1 wird nun mit zwei Eingaben aufgerufen. Die erste Eingabe wird unter dem Namen :LAENGE und die zweite unter :BREITE abgelegt. In den Zeilen (VORWAERTS :LAENGE) und (VORWAERTS :BREITE) werden nun die Linien mit den eingegebenen Längen gezeichnet, so daß jedes beliebige Rechteck gezeichnet werden kann.

Alle Eingaben, die in der Titelzeile für die Prozedur festgelegt sind, sind "Privatbesitz" dieser Prozedur. Diese Namen sind außerhalb der Prozedur nicht bekannt.

2.6 Ausdrucken von Prozeduren

Wir können uns eine Prozedur auch vom Befehlsmodus aus "zeigen" lassen. Hierzu dient der Befehl ZEIGE (vgl. auch Kap. 5.1).

```
?ZEIGE "RECHTECK
PR RECHTECK
 VORWAERTS 50
 RECHTS 90
 VORWAERTS 25
 RECHTS 90
 VORWAERTS 50
 RECHTS 90
 VORWAERTS 25
 RECHTS 90
ENDE
?...
```

Im Unterschied zu den Befehlen EDIT und LERNE muß der Name der zu zeigenden Prozedur wie ein Wort mit Anführungszeichen eingegeben werden.
Überhaupt sind "EDIT" und "LERNE" die einzigen LOGO-Befehle, wo der Name als Eingabe nicht von Anführungszeichen angeführt werden muß. Denn bei diesen Befehlen ist es ja nicht sehr sinnvoll, den Namen der Prozedur, die zu lernen oder zu editieren ist, erst vorher zu berechnen.
Würden wir nun bei ZEIGE den Namen ohne Anführungszeichen eingeben, so würde LOGO erst die zu zeigende Prozedur ausführen und sich daraus ein Ergebnis erhoffen (welches dann der gesuchte Prozedurname sein sollte). Durch die Anführungszeichen wird also die Ausführung der Prozedur verhindert, und das Wort direkt als Eingabe genommen.

```
?ZEIGE RECHTECK                    ; es wird ein Rechteck gezeichnet, dann ...
 Keine Rückgabe von : RECHTECK
?...
```

An dieser Stelle sehen wir auch, daß eine LOGO-Prozedur durchaus ein Ergebnis zurückgeben kann.

2.7 Prozeduren mit Ergebnissen (Rückgaben)

Wir wollen nun eine Prozedur KREISUMFANG definieren, die die Länge des Kreisumfangs aus dem Radius berechnet. Die Formel hierzu lautet :

KREISUMFANG = 2 * Pi * RADIUS

Die Prozedur KREISUMFANG soll also das Ergebnis dieser Formel zurückgeben. Der Befehl dazu ist

- RUECKGABE (kurz: RG).

Die Prozedur KREISUMFANG müßte also lauten:

```
PR KREISUMFANG :RADIUS
 RUECKGABE 2 * 3.1415 * :RADIUS
ENDE
```

In dem Moment, wo AIP - LOGO auf den Befehl RUECKGABE trifft, bricht es die Bearbeitung der Prozedur ab, und gibt die Eingabe zum Befehl RUECKGABE (hier : 2 * 3.1415 * :RADIUS) als Ergebnis zurück.

```
?DRUCKEZEILE KREISUMFANG 2
12.566
?...
```

Das Ergebnis der Prozedur KREISUMFANG mit der Eingabe 2 ist 12.566. Dieses Ergebnis wird nun als Eingabe für DRUCKEZEILE genommen. So druckt LOGO 12.566 aus.

Wenn nun nicht gesagt wird, was mit dem Ergebnis einer Prozedur gemacht werden soll, dann meldet AIP - LOGO :

```
?KREISUMFANG 2
 Ergebnis : 12.566
?...
```

Wenn dies nun aber innerhalb einer Befehlszeile oder in einer Prozedur geschieht, meldet AIP - LOGO einen Fehler:

```
?VORWAERTS 25 KREISUMFANG 2 RECHTS 90
Was soll geschehen mit 12.566
?...
```

Nach der Ausführung von VORWAERTS 25 wird die Prozedur KREISUMFANG bearbeitet. LOGO kann nun mit dem Ergebnis von KREISUMFANG nichts anfangen und meldet dies. Die Befehlszeile wird hier abgebrochen, und der Befehl RECHTS 90 wird nicht mehr ausgeführt.

Eine Prozedur kann auch ohne Rückgabe vorzeitig abgebrochen werden.

- RUECKKEHR (kurz: RK)

Der Befehl RUECKKEHR veranlaßt LOGO, die Ausführung der Prozedur sofort zu verlassen und ohne ein Ergebnis zurückzukehren.

```
PR VERSUCH
 DRUCKEZEILE 1
 RUECKKEHR
 DRUCKEZEILE 2
ENDE

?VERSUCH
1
?...
```

Die Prozedur VERSUCH wird in der zweiten Zeile vorzeitig abgebrochen, so daß der Befehl DRUCKEZEILE 2 in der dritten Zeile nicht mehr ausgeführt wird.

2.8 Macro Prozeduren

Wie in vielen Programmiersprachen, die in das Gebiet der KI (künstlichen Intelligenz) hineinragen, ist es auch in AIP - LOGO möglich, Macros zu definieren.

Macros unterscheiden sich von "normalen" Prozeduren nur in einem Punkt: Sie können immer nur mit einer Eingabenforderung versehen werden. Diese Eingabe wird dann aber nicht - wie gewohnt - mit dem Ergebnis des im Aufruf folgenden Begriffs besetzt, sondern mit der unausgewerteten Restzeile.

Dies wird am besten anhand einiger Beispiele deutlich:

"normale Prozedur"

```
PR TEST1 :VARIABLE
 DRUCKEZEILE :VARIABLE
ENDE

?TEST1 2 * 5
10
?TEST1 8 / 2 DRUCKEZEILE 8 / 4
4
2
?...
```

nun eine macro Prozedur (Macro):

```
PR TEST2 ::VARIABLE                          ; Beachte: zwei (!) Doppelpunkte
 DRUCKEZEILE :VARIABLE
ENDE

?TEST2 2 * 5
2 * 5
?TEST2 8 / 2 DRUCKEZEILE 8 / 4
8 / 2 DRUCKEZEILE 8 / 4
?...
```

Wie Sie sehen, wird die Restzeile zum Aufruf von TEST2 nicht wie gewohnt ausgewertet, sondern als "roher" Satz übergeben.
Beim Schreiben unterscheiden sich normale und macro Prozeduren nur durch den zweiten Doppelpunkt vor dem Eingabennamen in der Titelzeile. Der Eingabenname kann aber innerhalb der Prozedur wie gewohnt abgefragt werden.

3 Programmieren mit Prozeduren

Im folgenden werden einige Bespiele aus der Igelgrafik aufgeführt und erläutert. Anschließend (ab Kapitel 3.4) werden noch alle restlichen Befehle zur Igelgrafik angesprochen.

Bevor Sie die einzelnen Prozeduren ausprobieren, sollten Sie mit dem Befehl BILD (kurz: BI) in den Bildmodus gehen. Wenn Sie vom Befehlsmodus aus Igelbefehle aufrufen, meldet AIP - LOGO einen Fehler:

(im Befehlsmodus:)

?RECHTS 50
RECHTS ist nur im BILD-Modus erlaubt !
?...

Wenn Sie sich bereits im BILD-Modus befinden, sollten Sie vor jedem Aufrufen der folgenden Prozeduren den Befehl

- STARTIGEL (kurz: SI)

aufrufen. Dieser Befehl startet die Igelgrafik völlig neu. Alle Igeldaten werden neu gesetzt, und auch das Grafikbild wird gelöscht. Als einzigster Faktor wird die Skala (vgl. Kapitel 3.6) nicht wieder zurückgesetzt.

Sie können auch Teile der Igelgrafik neu starten:

- LOESCHEBILD (kurz: LB)

Dieser Befehl löscht nur das Grafikbild. Der Standort des Igels usw. wird nicht geändert.

- MITTE

Mit diesem Befehl können Sie den Igel in seine Ausgangsposition (in der Mitte) setzen. Das Bild wird dabei nicht verändert.

3.1 Eine Treppe

Eine Treppe besteht aus mehreren Stufen. Wir wollen zunächst eine Prozedur schaffen, die eine Stufe zeichnet.

```
PR STUFE1
 VORWAERTS 10
 RECHTS 90
 VORWAERTS 10
 LINKS 90
ENDE
```

Diese Prozedur STUFE1 zeichnet eine Stufe der Höhe und Breite 10. Wenn wir die Größe der Stufe jedesmal mitbestimmen wollen, müssen wir die Prozedur mit zwei Eingaben versehen:

```
PR STUFE :HOEHE :BREITE
 VORWAERTS :HOEHE
 RECHTS 90
 VORWAERTS :BREITE
 LINKS 90
ENDE
```

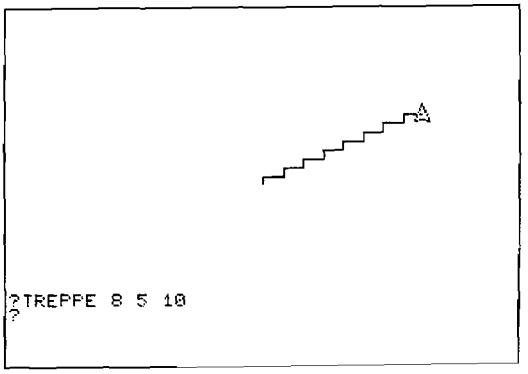
Nun wollen wir eine Prozedur schaffen, die eine Treppe mit verschiedener Stufenanzahl und verschiedenen Stufengrößen zeichnet:

```
PR TREPPE :STUFENZAHL :HOEHE :BREITE
 WIEDERHOLE :STUFENZAHL [STUFE :HOEHE :BREITE]
ENDE
```

```
?TREPPE 8 5 10
 ...
```

Dieser Befehl zeichnet jetzt eine Treppe mit 8 Stufen, die jeweils 5 Einheiten hoch und 10 Einheiten breit sind (s. Bild 3.1).

(Bild 3.1)



Wenn wir nun für die Treppe die Gesamtbreite und -höhe eingeben wollen, müssen wir die Prozedur TREPPE etwas abändern:

```
PR TREPPE1 :STUFEN :HOEHE :BREITE
 WH :STUFEN [STUFE (:HOEHE / :STUFEN) (:BREITE / :STUFEN)]
ENDE
```

Wenn wir nun

```
?TREPPE1 10 80 50
 ...
```

eingeben, so wird eine Treppe mit 10 Stufen gezeichnet, die eine gesamte Höhe von 80 und eine gesamte Breite von 50 Einheiten hat (s. Bild 3.2).
Wir wollen dieses Beispiel etwas näher betrachten. Die Prozedur TREPPE1 wird aufgerufen. Dabei erhalten die Namen für die Eingaben folgende Werte:

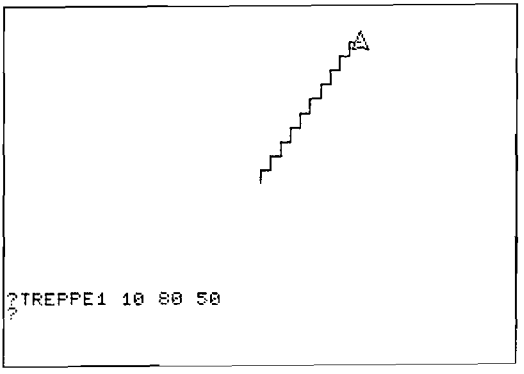
```
Prozedur TREPPE1:
     :STUFEN   = 10
     :HOEHE    = 80
     :BREITE   = 100
```

Nun wird die Prozedur STUFE zehnmal innerhalb der Prozedur TREPPE1 aufgerufen. Dort werden folgende Werte übergeben:

```
Prozedur STUFE:
     :HOEHE    = 80  / 10 =  8
     :BREITE   = 100 / 10 = 10
```

Die Namen :HOEHE und :BREITE haben also in den verschiedenen Prozeduren verschiedene Werte!
Dies ist nur möglich, weil die Namen und die ihnen zugeordneten Werte reiner Privatbesitz der einzelnen Prozeduren sind. Diese Namen der Eingaben sind ausschließlich innerhalb der betreffenden Prozedur bekannt.
Im Kapitel 4 wenden wir uns dann noch ausführlich den Namen - und den Möglichkeiten, mit Namen zu arbeiten - zu.

(Bild 3.2)



3.2 Kreise und Kreisbögen

Kreise und Kreisbögen (Kreisteile) sind Bestandteil vieler Zeichnungen. Es gibt in LOGO aber keine Befehle, die Kreise zeichnen. Wir wollen nun versuchen, einige Prozeduren zum Thema Kreis aufzustellen.

Es wäre natürlich möglich, jeden Punkt des Kreises über trigeometrische Funktionen (SIN, COS, TAN) auszurechnen. Hierauf wollen wir aber verzichten. Eine gute Näherung an einen Kreis ergibt sich schon, wenn der Igel immer wieder einen kleinen festen Schritt vorwärts macht und sich dann um einen sehr kleinen festen Winkel dreht. Wenn der Igel dann 360 Grad durchlaufen hat, ist der Kreis geschlossen.
Durch die Größe des Winkels, um den sich der Igel jedesmal dreht, wird die Genauigkeit des Kreises bestimmt. Das einfachste Beispiel ist, wenn sich der Igel jeweils um eine Einheit vorwärts bewegt und sich dann um ein Grad dreht. Dies muß er 360 mal wiederholen.
Die Prozedur KREIS1:

```
PR KREIS1
 WIEDERHOLE 360 [VORWAERS 1 RECHTS 1]
ENDE

?STARTIGEL
?KREIS1                                  ; im BILD-Modus !
 ...
```

Es wird ein Kreis gezeichnet. Diese Prozedur ist sehr langsam. Erheblich schneller ist sie schon, wenn wir vorher VERSTECKIGEL eingeben. Dann muß nicht ständig der Igel auf dem Bildschirm gesetzt und zurückgesetzt werden.

```
 ...
?LB                                      ; Abkürzung für LOESCHEBILD
?KREIS1
 ...
```

Trotzdem ist die Prozedur KREIS1 noch recht langsam. Es wird in ihr ja 360 mal die Befehlsfolge (VORWAERTS 1 RECHTS 1) ausgeführt. Es ist jedoch gar nicht notwendig, so genau zu zeichnen. Auch wenn sich der Igel immer gleich um 10 Einheiten vorwärts bewegt und dann um 10 Grad dreht, ist der Kreis noch sehr genau:

```
PR KREIS2
 WH 36 [VORWAERTS 10 RECHTS 10]
ENDE

?LB
?KREIS2
 ...
```

Diese Prozedur KREIS2 ist auch bei gezeigtem Igel schon recht schnell.

Allgemein brauchbar wird die Prozedur erst, wenn wir die Schrittweite, also damit die Größe des Kreises, eingeben können. Das sieht so aus:

```
PR KREIS3 :SCHRITT
 WH 36 [VW :SCHRITT RE 10]
ENDE
```

Wenn wir jetzt nicht die Schrittgröße eingeben wollen, sondern den Radius des Kreises, so müssen wir den Radius erst umrechnen:

```
 SCHRITT = 2 * Pi * RADIUS / 36
         = Pi / 18 * RADIUS
         = 0.174533 * RADIUS
```

Die Schrittweite ergibt sich also aus dem Produkt 0.174533 * RADIUS.

```
PR KREIS4 :RADIUS
 WH 36 [VW (:RADIUS * 0.174533) RE 10]
ENDE
```

Wir haben in der Wiederhole-Zeile :SCHRITT durch das Produkt (:RADIUS * 0.174533) ersetzt. Die Prozedur KREIS4 zeichnet einen Kreis mit dem eingegebenen RADIUS.
Noch besser wäre es, wenn der Kreis von seinem Mittelpunkt aus gezeichnet werden könnte. Dies wird in der Prozedur KREIS verwirklicht. Es wird hier erst vom Mittelpunkt zum Rand des Kreises gesprungen (mit STFTHOCH), dann der Kreis gezeichnet und schließlich wieder in die Mitte des Kreises gesprungen.
Es in dieser Prozedur KREIS noch eine weitere Feinheit eingefügt worden, um den Kreis ganz genau um die Igelposition zu zeichnen. Um die Prozedur noch etwas schneller zu machen, könnte man noch den Befehl VERSTECKIGEL (und am Ende der Prozedur ZEIGIGEL) aufrufen.

```
PR KREIS :RADIUS
 STIFTHOCH
 VORWAERTS :RADIUS
 STIFTAB
 RECHTS 90
 VW :RADIUS * 0.174533 / 2
 RE 10
 WH 35 [VW (:RADIUS * 0.174533) RE 10]
 VW :RADIUS * 0.174533 / 2
 LINKS 90
 STIFTHOCH
 RUECKWAERTS :RADIUS
 STIFTAB
ENDE
```

Die folgenden Zeilen sind als kleine Korrektur zu verstehen. Außerdem muß noch für den WIEDERHOLE-Befehl 35 eingesetzt werden.

```
VW :RADIUS * 0.174533 / 2
RE 10
VW :RADIUS * 0.174533 / 2
```

Häufig braucht man nur einen Teil eines Kreises - einen Kreisbogen. Für diesen Kreisbogen müssen wir dann nicht nur einen Radius angeben, sondern auch noch den Winkel, den er überstreichen soll. Hier bietet sich folgende einfache Möglichkeit:

```
PR KREISBOGEN1 :RADIUS :WINKEL
 WH (:WINKEL / 10) [VW (:RADIUS * 0.174533) RE 10]
ENDE
```

Achtung:
Die Eingabe für den Winkel muß immer durch zehn teilbar sein.

Die Prozedur KREISBOGEN1 macht immer genau soviele Schritte, wie der zu durchlaufende Winkel (durch zehn!) groß ist. Dabei bleibt die Schrittgröße nur vom eingegebenen Radius abhängig.

Wir wollen jetzt eine Prozedur KREISBOGEN2 definieren, die als Eingabe einen beliebigen Winkel bekommen kann.

```
PR KREISBOGEN2 :RADIUS :WINKEL
 WH 36 [VW (:RADIUS * 0.174533 * :WINKEL / 360) RE (:WINKEL / 36)
ENDE
```

Etwas genauer wird die Prozedur noch, wenn wir die kleine Korrektur, die wir auch in der Prozedur KREIS gemacht haben, einfügen:

```
PR KREISBOGEN :RADIUS :WINKEL
 VW (:RADIUS * 0.174533 * :WINKEL / 360 / 2)
 RE (:WINKEL / 36)
 WH 35 [VW (:RADIUS * 0.174533 * :WINKEL / 360) RE (:WINKEL / 36)]
 VW (:RADIUS * 0.174533 * :WINKEL / 360 / 2)
ENDE
```

Diese Prozedur KREISBOGEN macht nun immer eine konstante Anzahl von Schritten. Dies ist hier sinnvoll, da wir umso genauer zeichnen müssen, je kleiner der Winkel ist.

Die Prozedur KREISBOGEN zeichnet nun immer Kreisbögen nach rechts - im Uhrzeigersinn. Wir wollen diese Prozedur jetzt RECHTSBOGEN nennen. Wir definieren uns dann noch die Prozedur LINKSBOGEN, in der (als Veränderung zu RECHTSBOGEN) nur RECHTS durch LINKS ersetzt werden muß:

```
PR LINKSBOGEN :RADIUS :WINKEL
 VW (:RADIUS * :WINKEL * 0.174533 / 360 / 2)
 LINKS (:WINKEL / 36)
 WH 35 [VW (:RADIUS * :WINKEL * 0.174533 /360) LI (:WINKEL / 36)]
 VW (:RADIUS * :WINKEL * 0.174533 / 360 / 2)
ENDE
```

Es gibt noch eine zweite Möglichkeit, die Prozedur LINKSBOGEN zu definieren:

```
PR LINKSBOGEN1 :RADIUS :WINKEL
 RECHTSBOGEN (- :RADIUS) (- :WINKEL)
ENDE
```

Diese Prozedur LINKSBOGEN1 nutzt die Eigenschaft des Befehls RECHTS - daß seine Eingabe auch negativ sein kann - aus. Ebenso können auch die Befehle LINKS, VORWAERTS und RUECKWARTS mit negativen Zahlen aufgerufen werden.

Noch ein Hinweis:
Wenn Ihnen die Prozeduren RECHTSBOGEN und LINKSBOGEN noch zu langsam sind, so können Sie die folgenden erheblich schnelleren Prozeduren RBOGEN und LBOGEN benutzen. Diese beiden Prozeduren sind jedoch etwas ungenauer!

```
PR RBOGEN :R :W
 BOGEN (:R * :W * 0.000969628) (:W / 18)
ENDE

PR LBOGEN :R :W
 BOGEN (:R * :W * 0.000969628) (- :W / 18)
ENDE

PR BOGEN :SCHRITT :TEILWINKEL
 VW (:SCHRITT / 2)
 RE :TEILWINKEL
 WH 17 [VW :SCHRITT RE :TEILWINKEL]
 VW (:SCHRITT / 2)
ENDE
```

Es bestehen zwei Unterschiede zu den Prozeduren RECHTS- und LINKSBOGEN. Zum einen werden nur noch 18 und nicht mehr 36 Schritte ausgeführt. Zum anderen aber werden die Eingaben in der Wiederholezeile nicht mehr jedesmal neu berechnet. Das macht diese Prozeduren erheblich schneller.

Außerdem können Sie noch den Igel verstecken!

Wir wollen nun unsere Prozeduren RECHTSBOGEN und LINKSBOGEN (bzw. auch RBOGEN und LBOGEN) zum Zeichnen einer Blume nutzen.

```
PR HALBBLATT :GROESSE
 RECHTSBOGEN :GROESSE 60
 RECHTS 120
 RECHTSBOGEN :GROESSE 60
 RECHTS 120
ENDE

PR BLUME :GROESSE
 WH 6 [HALBBLATT :GROESSE RE 60]
ENDE

PR BLUME1 :GROESSE
 WH 16 [RECHTSBOGEN :GROESSE 48 RE 120]
ENDE
```

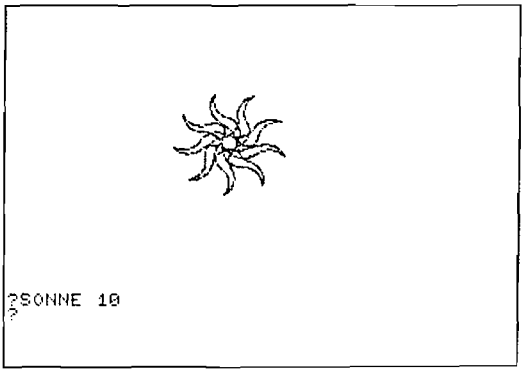
Die folgende Prozedur WELLE zeichet eine Welle, bestehend aus vier sich entgegengesetzt abwechselnden Kreisbögen. Wenn wir viele dieser Wellen immer etwas verdreht zeichnen, entsteht eine Sonne:

```
PR WELLE :GROESSE
 LINKSBOGEN :GROESSE 90
 RECHTSBOGEN :GROESSE 90
 LINKSBOGEN :GROESSE 90
 RECHTSBOGEN :GROESSE 90
ENDE

PR SONNE :GROESSE
 WH 9 [WELLE :GROESSE RECHTS 160]
ENDE
```

Ein durch die Prozedur SONNE erzeugtes Bild sehen Sie in Bild 3.3.
Die Idee zur dieser Prozedur hat der Autor dem Buch "Einführung in LOGO" (siehe Literaturhinweis <1>, Anhang VI) entnommen. Es eignet sich wegen der vielen Beispielprogramme hervorragend als Ergänzung zu diesem Handbuch.

Nun sind Sie an der Reihe, sich Bilder zu überlegen, und diese mit Hilfe der Prozeduren RECHTSBOGEN und LINKSBOGEN zeichnen zu lassen.



(BILD 3.3)

3.3 Sich wiederholende Prozeduren

Häufig entstehen sehr hübsche Figuren, wenn wir eine Prozedur schaffen, die sich selbst immer wiederholt. Das beste Beispiel hierzu ist die Prozedur VIELECK, die solange Linien zeichnet (immer um einen festen Winkel versetzt), bis sie durch "Ctrl-C" abgebrochen wird.

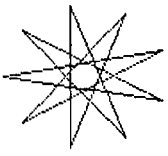
```
PR VIELECK :GROESSE :WINKEL
 VORWAERTS :GROESSE
 RECHTS :WINKEL
 VIELECK :GROESSE :WINKEL
ENDE
```

Einige Figuren, die von VIELECK erzeugt wurden, werden in Bild 3.4 gezeigt. Es wurde dabei jedesmal, dachdem sich die Prozedur geschlossen hatte, mit "Ctrl-C" abgebrochen.

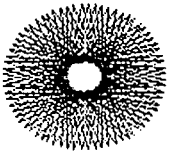
Es wäre ja wünschenswert, daß sich die Prozedur VIELECK selber stoppt, wenn sich die Figur geschlossen hat. Die Möglichkeiten, dieses zu erreichen, werden wir in Kapitel 3.3.1 kennenlernen.

Wenn sich eine Prozedur selber aufruft, sprechen wir grundsätzlich von Rekursion. Sinnvoll ist eine Rekursion aber meist nur, wenn sich die Prozedur aufgrund eines Ereignisses - z.B., daß ein Zähler auf null herabgezählt ist - selber abbrechen kann. Dazu müssen wir die Fallunterscheidung einführen.

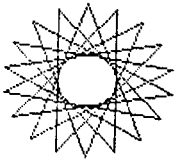
(Bild 3.4)



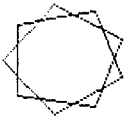
VIELECK 80 160



VIELECK 80 155



VIELECK 80 140



VIELECK 40 80

3.3.1 Bedingte Entscheidung : WENN...DANN...SONST

Wir wollen eine einfache Prozedur definieren, die zählen kann.
Dazu muß sich die Prozedur immer wieder selbst mit einer um eins erhöhten Eingabe aufrufen.

```
PR ZAEHLE :ZAHL
 DZ :ZAHL
 ZAEHLE :ZAHL + 1
ENDE
```

```
?ZAEHLE 0
0
1
2
3
 ...
```

Die Prozedur muß mit "Ctrl-C" abgebrochen werden.

Nun wollen wir eine Prozedur definieren, die eine Zahl herunterzählt. Wenn die Zahl null ist, soll sich die Prozedur selbstständig abbrechen.
Wir müssen zunächst die Bedingung für den Abbruch erstellen:

```
WENN :ZAEHLER = 0 DANN RUECKKEHR
```

Diese Abbruchbedingung fügen wir nun in die Prozedur ein.

```
PR ZAEHLE1 :ZAEHLER
 DZ :ZAEHLER
 WENN :ZAEHLER = 0 DANN RUECKKEHR
 ZAEHLE1 :ZAEHLER - 1
ENDE
```

```
?ZAEHLE1 3
3
2
1
0
?...
```

Um die Wirkung von dem Befehl SONST zu zeigen, können wir die Prozedur ZAEHLE auch so schreiben:

```
PR ZAEHLE2 :ZAHL
 DZ :ZAHL
 WENN :ZAHL = 0 DANN RK SONST ZAEHLE2 :ZAHL - 1
ENDE
```

Der Ausdruck

```
:ZAHL = 0
```

ist eigentlich eine Frage - die Frage danach, ob :ZAHL den Wert null hat. Und so antwortet AIP - LOGO auch - mit WAHR oder FALSCH. Wir können solche Fragen auch direkt eingeben:

```
?12 = 45
 Ergebnis : FALSCH
?3 = 3
 Ergebnis : WAHR
?...
```

"=" prüft die Gleichheit zweier Eingaben bzw. ob zwei Zahlen den gleichen Wert haben.
Genauso prüfen folgende Symbole, ob die eine Zahl größer, kleiner, größergleich, kleinergleich oder ungleich einer zweiten ist:

```
> , < , >= , <= , <>
```

Der Befehl WENN verlangt als Eingabe genau diese Antwort auf eine Frage - also ein WAHR oder FALSCH. Wir könnten statt dieser Frage auch gleich eine Antwort eingeben. Dies ist aber wenig sinnvoll:

```
?WENN WAHR DANN DRUCKZEILE 1 SONST DRUCKZEILE 2
1
?...
```

Hier findet eigentlich keine Unterscheidung mehr statt, da die Bedingung WAHR ja immer wahr ist.

Sinnvoll dagegen kann es sein, eine Prozedur zu schaffen, die als Ergebnis WAHR oder FALSCH hat. Diese Prozedur könnte dann als Frage aufgerufen werden. Es ist ratsam, solche Prozeduren, die als Ergebnis WAHR oder FALSCH haben sollen, mit einem Fragezeichen am Ende zu benennen. Man erkennt dann sofort den fragenden Charakter.

```
PR GLEICHEINS? :ZAHL
 RUECKGABE :ZAHL = 1
ENDE
```

Diese Prozedur gibt als Ergebnis WAHR oder FALSCH zurück, je nachdem, ob die eingegebene Zahl gleich eins ist oder nicht. Es gibt in AIP - LOGO auch einige Grundbefehle, die Eingaben auf bestimmte Eigenschaften hin untersuchen, und als Ergebnis WAHR oder FALSCH liefern. Alle diese LOGO-Fragen haben ein Fragezeichen am Ende.
Z.B.:

```
?PROZEDUR? "GLEICHEINS?
 Ergebnis : WAHR
?PROZEDUR? "DIESEPROZEDURGIBTESNICHT
 Ergebnis = FALSCH
 ...
```

Die Frage PROZEDUR? prüft, ob es sich bei der Eingabe um den Namen einer Prozedur handelt. Weitere solche LOGO-Grundbefehle (-fragen) werden wir noch kennenlernen.

3.3.2 Logische Befehle von AIP - LOGO

Es gibt in LOGO einige rein logische Befehle.

- NICHT

Dieser Befehl liefert als Ergebnis das logische Gegenteil:

```
?NICHT WAHR
 Egrebnis = FALSCH
?DZ NICHT FALSCH
WAHR
?...
```

- EINES?

Die LOGO-Frage EINES? entspricht dem logischen "ODER". Sie fragt, ob eine der beiden Eingaben (beides müssen Wahrheitswerte - also WAHR oder FALSCH - sein) WAHR ist. Wenn ja wird WAHR, wenn nein, wird FALSCH zurueckgegeben. Die Frage ZAHL? prüft, ob die Eingabe eine Zahl ist.

```
?EINES? ZAHL? 2 ZAHL? "ANJA
 Ergebnis = WAHR
?DZ EINES? ZAHL? "ANJA FALSCH
FALSCH
?...
```

- ALLE?

Diese LOGO-Frage entspricht dem logischen "UND". Nur wenn beide Eingaben WAHR sind, wird WAHR zurückgegeben, sonst FALSCH.

```
?DZ ALLE? ZAHL? 2 ZAHL? "ANJA
FALSCH
?DZ ALLE? ZAHL? 2 ZAHL? 0
WAHR
?...
```

Auch diese beiden logischen Fragen können in AIP - LOGO mit beliebig vielen Eingaben aufgerufen werden - in runden Klammern selbstverständlich. Es wird dann jeweils gefragt, ob eine der Eingaben WAHR ist, bzw., ob alle Eingaben WAHR sind. So ist es logisch, daß die Frage EINES? ohne eine Eingabe FALSCH ergeben muß, während die Frage ALLE? ohne Eingaben WAHR als Antwort hat.

```
?DZ (EINES? 1 = 2  2 = 3  5 = 5)
WAHR
?DZ (ALLE? 1 = 1  2 = 2  5 = 7)
FALSCH
?...
```

3.3.3 TESTE... WENNWAHR... WENNFALSCH...

- TESTE

Der Befehl TESTE hat in LOGO die gleiche Aufgabe, wie die Befehlsfolge WENN...DANN...SONST. Er steht immer in Verbindung mit mindestens einem der Befehle WENNWAHR und WENNFALSCH. TESTE testet, ob etwas WAHR oder FALSCH ist. AIP - LOGO merkt sich das Ergebnis bis zum erneuten Aufruf von TESTE.

- WENNWAHR (kurz: WW)
- WENNFALSCH (kurz: WF)

Die Restzeile hinter dem Befehl WENNWAHR wird nur ausgeführt, wenn das letzte Prüfergebnis WAHR war. Ebenso wird die Restzeile hinter WENNFALSCH nur ausgeführt, wenn das letzte Testergebnis FALSCH war. Die Befehle WENNWAHR und WENNFALSCH beziehen sich also immer auf den letzten Aufruf von TESTE.

Die Prozedur ZAEHLE3 könnte mit dem Befehl TESTE so aussehen:

```
PR ZAEHLE3 :ZAHL
 DRUCKZEILE :ZAHL
 TESTE :ZAHL = 0
 WENNFALSCH ZAEHLE :ZAHL - 1
ENDE
```

Es ist möglich, das Prüfergebnis von TESTE mehrmals zu verwenden. Dieses Ergebnis bleibt bis zum nächsten TESTE-Aufruf erhalten. Dies gilt auch, wenn sich die Abfrage (WENNWAHR/-FALSCH) in einer anderen Prozedur befindet.

Nun zu einer weiteren Programmiertechnik:
Wir können die Prozedur ZAEHLE4 auch "rückwärts-rekursiv" schreiben. Das bedeutet, daß - trotz der Bearbeitung der Zahlen rückwärts - die Zahlen vorwärts ausgedruckt werden. Dies klingt sehr trocken. Doch wenden wir uns einem Beispiel zu:

```
PR ZAEHLE4 :ZAHL
 WENN :ZAHL > 0 DANN ZAEHLE4 :ZAHL - 1
 DZ :ZAHL
ENDE

?ZAEHLE4 4
0
1
2
3
4
?...
```

Wie stellen etwas Erstaunliches fest:
Obwohl die Zahlen von 4 nach 0 abgearbeitet werden, werden sie umgekehrt von 0 nach 4 ausgegeben. Dies erklärt sich folgendermaßen:
Die Prozedur ZAEHLE3 druckt erst die Zahl aus, und ruft sich dann selbst mit einer um eins kleineren Zahl auf. Die Prozedur ZAEHLE4 arbeitet mit umgekehrter Reihenfolge. Sie prüft erst, ob sie sich selbst nochmal aufrufen muß, tut dieses wenn nötig, und druckt erst dann die Zahl aus.
Diese Technik wird manchmal auch rekursiv-nachklappernd genannt, weil sich die Prozedur erst bis auf die tiefste Ebene herunterzählt, und dann beim "Wiederhochkommen" die Zahlen "nachklappernd" ausdruckt.

Ganz erstaunliche Figuren liefern auch Prozeduren, die sich selbst gleich mehrmals aufrufen. Folgende Prozedur MUSTER ruft sich zweimal selbst auf:

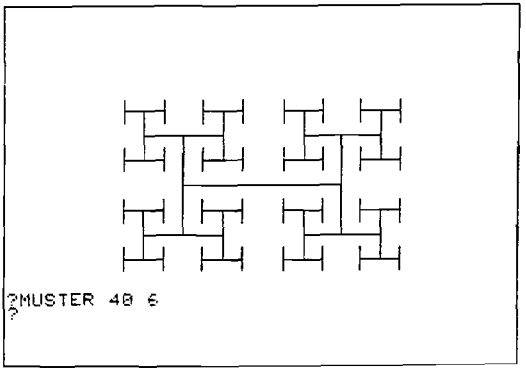
```
PR MUSTER :GROESSE :TIEFE
 VERSTECKIGEL
 WENN :TIEFE = 0 DANN RUECKKEHR
 RECHTS 90
 VORWAERTS :GROESSE
 MUSTER (:GROESSE * 0.7) (:TIEFE - 1)
 RUECKWAERTS :GROESSE * 2
 MUSTER (:GROESSE * 0.7) (:TIEFE - 1)
 VORWAERTS :GROESSE
 LINKS 90
ENDE
```

```
?BILD
?MUSTER 40 6
 ...
```

Der Aufruf der Prozedur MUSTER mit der Größe 40 und der Tiefe 6 erzeugt das Bild 3.5.

Versuchen Sie auch mal die Größe 60 und die Tiefe 10. Sie sollten dann aber mit "Ctrl-B" in den Vollbildmodus übergehen, um das ganze Bild sehen zu können.

(Bild 3.5)



3.4 Das Koordinatenfeld des Igels

Das Koordinatenfeld des Igels hat seinen Urspung (Nullpunkt) in der unteren linken Ecke des Bildschirms. Die y-Achse ist in 200 Einheiten unterteilt. Die Einteilung der x-Achse hängt von der eingegebenen Skala (s. Kap. 3.6) ab. Nach Start von AIP - LOGO ist die x-Achse in 256 Einheiten unterteilt. Das Bild 3.6 zeigt eine Abbildung des Igel-Koordinatenfeldes in AIP - LOGO.

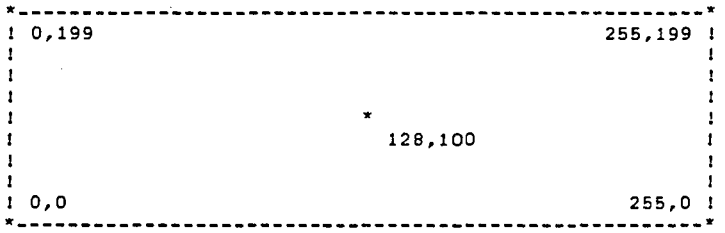
Nun sollen Befehle vorgestellt werden, die den Igel innerhalb dieses Koordinatenkreuzes bewegen. Als erstes sind da die Befehle:

- AUFXY
- AUFX
- AUFY

Wenn wir bisher den Igel bewegen wollten, mußten wir immer die Befehle VORWAERTS und RUECKWAERTS benutzen. Diese Befehle bewegen den Igel jedoch immer relativ zu seinem Standort. Wenn wir den Igel nun aber unabhängig von seinem Standort - auf einen bestimmten Punkt zu - bewegen wollen, so können wir den Befehl AUFXY nutzen. AUFXY benötigt zwei Eingaben. Die X- und die Y-Koordinate des Punktes, wohin sich der Igel bewegen soll.

(Bild 3.6)

Das "Igelkoordinatenfeld" :



Die Angaben für die X-Koordinate verändern sich bei einer neuen Skala!

Nach dem Start von AIP - LOGO - oder nach dem Aufruf von STARTIGEL oder MITTE (selbstverständlich alles im Bildmodus!) - steht der Igel immer in der Bildschirmmitte auf folgendem Punkt: 128,100.
Wir befehlen nun:

?STARTIGEL
?AUFXY 150 150
?...

Der Igel bewegt sich zeichnend von der Mitte nach oben rechts zum Punkt (150,150). Wichtig zu wissen ist, daß er dabei die Richtung seiner Nase n i c h t ändert. Er bewegt sich also hier nicht unbedingt in Richtung seiner Nase!

Entsprechend arbeiten auch die Befehle AUFX und AUFY:
Der Igel bewegt sich zu der jeweiligen X- oder Y-Koordinate. Dabei bleibt die andere Koordinate jeweils konstant.

```
?MITTE
?AUFX 200
?...
```

Der Igel bewegt sich um 72 (200 - 128) Schritte nach rechts zur X-Koordinate 200. Auch hierbei verändert sich die Richtung seiner Nase nicht.

Der Igel kann auch nach seinem aktuellen Standort befragt werden:

- XKO
- YKO

Diese beiden Befehle brauchen keine Eingaben und geben die jeweilige aktuelle Position (Koordinate) des Igels zurück.

```
?MITTE
?XKO
 Ergebnis = 128
?YKO
 Ergebnis = 100
?...
```

Auch die Richtung der Nase des Igels konnten wir bisher nur relativ (mit RECHTS und LINKS) verändern. Hier gibt es jetzt zwei Möglichkeiten, die Richtung absolut zu bestimmen:

- RICHTUNG (kurz: RI)
- AUFKURS (kurz: AK)

Durch den Befehl RICHTUNG mit Eingabe eines absoluten Winkels können wir die Nase des Igel in genau diese Richtung drehen.

Dabei ist:	0	= oben	(Norden)
	90	= rechts	(Osten)
	180	= unten	(Süden)
	270	= links	(Westen)

Natürlich können auch alle dazwischenliegenden Winkel eingegeben werden! Winkel über 360 Grad oder kleiner als null Grad werden in die entsprechenden Winkel zwischen 0 und 360 Grad umgewandelt (z.B. 450 wird zu 90 (450 = 90 + 360)).

Es gibt auch eine Möglichkeit, die Nase des Igels auf einen bestimmten Kurs zu bringen. Wir können erreichen, daß die Nase auf einen von uns bestimmten Punkt im Koordinatenfeld zeigt. Der Befehl heißt AUFKURS und braucht die beiden Koordinaten des anzupeilenden Punktes als Eingaben.

```
?STARTIGEL
?AUFKURS 200 100
?...
```

Der Igel zeigt von der Mitte aus nach rechts auf den Punkt (200,100).

Das ergibt den absoluten Winkel von 90 Grad (von oben aus gesehen). Wenn wir den Befehl RICHTUNG hätten benutzen wollen, so hätten wir
RICHTUNG 90
eingeben muessen.

Wir können nun auch die Richtung (den Kurs) der Nase des Igels abfragen:

- KURS

Dieser Befehl KURS liefert als Ergebnis den absoluten Winkel (s. RICHTUNG):

```
?RICHTUNG 90
?DZ KURS
90
?...
```

Wir können mit den Befehlen AUFKURS und VORWAERTS auch ohne Benutzung des Befehls AUFXY einen bestimmten Punkt ansteuern. Mit dem Befehl

- ENTFERNUNG (kurz: EF)

```
?STARTIGEL
?AUFKURS 200 150
?ENTFERNUNG 200 150
Ergebnis = 87.658428
?VORWAERTS 87.658428
?XKO
 Ergebnis = 200
?YKO
 Ergebnis = 150
?...
```

Der Befehl ENTFERNUNG gibt also die Entfernung des Igels von einem bestimmten Punkt an. ENTFERNUNG braucht zwei Eingaben, und zwar die Koordinaten des Punktes.

3.5 Weitere Kontrollfunktionen des Igels

Wenn wir versuchen, mit dem Igel über den Rand des Bildschirms hinauszugehen, erhalten wir eine Fehlermeldung.
Z.B.:

```
?MITTE
?VORWAERTS 150
 Igel ist im Aus !
?...
```

Es ist normalerweise nicht erlaubt, mit dem Igel den Bildschirm zu verlassen. Dies können wir mit dem Befehl

- RANDSPRUNG

ändern. Es wird dann ein "Sprung" über den Rand des Bildschirms zugelassen:

```
?SI                                    ; Abkürzung für: STARTIGEL
?RECHTS 90
?RANDSPRUNG
?VORWAERTS 150
?...
```

Jetzt wird der Befehl VORWAERTS 150 ausgeführt. Der Igel springt über den Rand des Bildschirms. Der Igel läuft an der genau gegenüberliegenden Seite des Bildschirms in der gleichen Richtung weiter (s. Bild 3.7). Wenn er den Bildschirm rechts verläßt, erscheint er in gleicher Höhe links wieder; wenn er oben verschwindet, erscheint er unten wieder. Wir können den Befehl RANDSPRUNG mit dem Befehl

- RAND

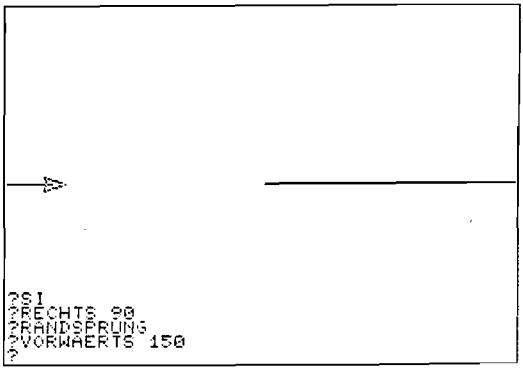
wieder rückgängig machen. Es wird dann wieder jeder Sprung über den Bildschirmrand verweigert und stattdessen die uns bekannte Fehlermeldung ausgegeben. Der Igel sieht den Rand dann wieder als eine unüberwindliche Grenze.

Den Befehl FARBE haben wir ja schon im ersten Kapitel besprochen. Mit seiner Hilfe können wir die Farbe des Schreibstiftes wählen. Nun gibt es einen weiteren Befehl, durch den wir die Farbe des unbemalten Hintergrundes bestimmen können.

- HINTERGRUND (kurz: HG)

Normalerweise ist im Bildmodus der Hintergrund schwarz. Wir können jetzt durch Aufruf des Befehls HINTERGRUND mit Eingabe einer Zahl die Farbe des Hintergrundes frei bestimmen. Die den einzelnen Farben entsprechenden Codezahlen finden Sie in der Tabelle (A.1) im Anhang.

(BILD 3.7)



Wenn wir z.B.

?HINTERGRUND 4

eingeben, so wird die Farbe des Hintergrundes von 0 auf 4 geändert. So kann auch die Farbe des Randes - das ist der Randbereich des Bildschirms, der nicht bemalt werden kann - geändert werden. Dazu ist folgender Befehl zu verwenden:

- RANDFARBE (kurz: RF)

?RANDFARBE 3

Der Aufruf von RANDFARBE mit der Eingabe "3" ändert die Farbe des Bildschirmrandes von schwarz auf Farbe 3 (s. Tab. (A.1) im Anhang).

Da die beiden folgenden Befehle (PALLETTE und PALLETTENEU) nicht auf allen Rechnertypen benutzt werden können, sehen Sie bitte im Anhang unterhalb der Tabelle (A.1) nach, ob diese Befehle bei Ihrem Rechner zugelassen sind!

- PALETTE (kurz: PAL)
- PALETTENEU (kurz: PALN)

Es stehen - je nach Rechnertyp und Zeichenmodus - eine bestimmte Anzahl von Farbstiften zur Verfügung. Genaueres entnehmen Sie bitte wieder dem Anhang Teil I.
Dabei ist zu beachten, daß der Farbstift mit der Nummer 0 immer die Farbe des Hintergrundes hat.

Die anderen Farbstifte können mit beliebigen Farben besetzt werden. Dazu wird der Befehl PALETTE benutzt. Dieser Befehl braucht zwei Eingaben. Die erste Eingabe muß die Nummer des Farbstiftes sein und die zweite die Codenummer der zuzuordnenden Farbe. Diese Codenummern stehen ebenfalls in der Tabelle (A.1) im Anhang.

Machen Sie im Bildmodus Versuch:

```
?FARBE 1
?WIEDERHOLE 4 [VW 10 RE 90]
?PALETTE 1 6
?...
```

Es wird ein Quadrat gezeichnet. Das Quadrat wird mit dem Farbstift 1 gezeichnet. Durch den Befehl PALETTE mit den Eingaben 1 und 6 wird die Farbe des Farbstiftes - und damit auch die Farbe des Quadrats - auf Farbe 6 (siehe Tab. (A.1) im Anhang) geändert.

Durch den Befehl PALLETTENEU werden alle Farbstifte, der Hintergrund und auch die Randfarbe auf die Werte, die sie schon bei Start des Systems hatten, gesetzt. Auch die Invertierung wird abgeschaltet (s.u.).

Es gibt in AIP - LOGO auch die Möglichkeit für den Igel seine Linien invertiert zu zeichnen. Das bedeutet, daß die Linien immer die entgegengesetzte Farbe zum Untergrund haben.

- INVERSEIN (kurz: IE)
- INVERSAUS (kurz: IA)

Die Invertierung wird mit dem Befehl INVERSEIN aktiviert, und mit dem Befehl INVERSAUS - oder auch mit dem Befehl PALETTENEU - wieder abgeschaltet.

Hierzu ein Beispiel:

```
?STARTIGEL
?FARBE 1
?WIEDERHOLE 4 [VW 20 RE 90]
?INVERSEIN
?RE 20 VW 50
?...
```

Wie Sie sehen, wird die Linie weiß gezeichnet, aber am Schnittpunkt der letzten Linie mit dem zuvor gezeichnete Quadrat entsteht die umgekehrte Farbe zum Untergrund (dies ist auf einem normalen Ferseher aber nur schwer zu erkennen!). Der Untergrund war mit Farbstift 1 gezeichnet, das heißt, daß die Farbe des Schnittpunktes jetzt dem umgekehrten Farbstift entspricht. Der sich ergebene Farbstiftwert läßt sich folgendermaßen berechnen:

Ergebnisfarbstiftwert bei Invertierung =
höchster erlaubter Farbstiftwert - Untergrundfarbstiftwert.

Alle in diesem Kapitel 3.5 angesprochenen Igelsteuerbefehle können auch vom Igel abgefragt werden. Es ist möglich, den Zustand des Igels zu erfragen. Dazu benötigen wir folgenden Befehl:

- IGELZUSTAND (kurz: IZ)

Ein Beispiel:

```
?STARTIGEL
?FARBE 1
?HG 3
?RANDFARBE 7
?RANDSPRUNG
?VERSTECKIGEL
?STIFTHOCH
?IGELZUSTAND
 Ergebnis = (1 3 7 FALSCH FALSCH WAHR FALSCH)
?...
```

Von dem Befehl IGELZUSTAND wird eine Liste zurückgegeben, die sieben Elemente enthält. Diese Elemente haben folgende Bedeutung:

(Tabelle 3.2)

1.	Element	aktuelle Farbstiftnummer
2.	Element	aktueller Hintergrundfarbwert
3.	Element	aktueller Randfarbenwert
4.	Element	WAHR oder FALSCH für STIFTAB oder STIFTHOCH
5.	Element	WAHR oder FALSCH für Igel ist zu sehen oder nicht
6.	Element	WAHR oder FALSCH für RANDSPRUNG oder RAND
7.	Element	WAHR oder FALSCH für Invertierung ein oder aus

Nun können Sie sich auch kleinere Prozeduren schreiben, die nur Teil des Igelzustandes abfragen. Folgende kleine Prozedur liefert immer WAHR, wenn der Igel zu sehen ist, und FALSCH, wenn er nicht zu sehen ist:

```
PR ISTIGELDA?
 RG (PICKE 5 IGELZUSTAND)
ENDE
```

Hierbei wird der Befehl PICKE benutzt. Er liefert als Ergebnis das n. Element (hier das 5.) einer Liste zurück (s. auch Kap. 6.4.3).

Es gibt in AIP - LOGO noch zwei weitere Befehle, die allerdings nicht typisch für diese Programmiersprache sind:

- FUELLE
- MALE

Der Befehl FUELLE wird ohne Eingaben aufgerufen. Der Igel fuellt von seinem Standort aus eine Fläche mit dem aktuellen Farbstift. Dabei muß die Fläche vollständig umschlossen sein. Ein Beispiel:

```
?STARTIGEL
?WIEDERHOLE 4 [VW 50 RE 90]                    ; Zeile 1
?RECHTS 45 VORWAERTS 30                        ; Zeile 2
?FUELLE                                        ; Zeile 3
?...
```

Wir lassen den Igel ein Quadrat der Seitenlänge 50 zeichnen (Zeile 1). Dann lassen wir den Igel in das Quadrat laufen (Zeile 2). Nun befindet sich der Igel in einem Bereich, der vollständig umschlossen ist. Dieses ist die Voraussetzung für den Aufruf des Befehls FUELLE (Zeile 3). Wenn die Fläche nicht vollständig umschlossen ist, so "läuft die Farbe aus", d.h. es wird der ganze Bildschirm mit dem aktuellen Farbstift gefüllt.
Es ist mit diesem Befehl möglich, auch die kompliziertesten Figuren, die von verschiedenen Farben eingegrenzt werden auszufüllen. Ein schönes Beispiel ist folgender Stern:

```
PR STERN
 STARTIGEL
 FARBE 1
 VW 40
 WIEDERHOLE 18 [VW 40 RE 160 VW 40 LI 140]
 RW 40
 FUELLE
ENDE

?STERN
?...
```

Die Ausführung der Prozedur STERN dauert ein wenig. Sie können diese Prozedur etwas schneller machen, indem Sie noch den Befehl VERSTECKIGEL in die zweite Zeile einfügen.

Der zweite - eigendlich "LOGO-fremde" - Befehl heißt MALE. Mit diesem Befehl ist es möglich, auch auf dem - eigendlich nur dem Igel reservierten - Grafikfeld Buchstaben oder Sätze zu schreiben. Der Befehl braucht vier Eingaben:

```
?STARTIGEL
?MALE 2 2 0 "LOGO
?...
```

Durch diesen Aufruf von MALE wird das Wort LOGO von der Bildschirmmitte - der Igelposition - aus nach rechts mit dem aktuellen Farbstift "gemalt" (geschrieben). Die einzelnen Buchstaben sind allerdings genau zweimal so hoch und zweimal so breit wie die Buchstaben, die Sie im unteren Bildschirmteil eingeben. Die Richtung, in die der Igel zeigt, ist hierbei ohne Bedeutung.

Nun zu der Bedeutung der einzelnen Eingaben:

1. Eingabe	Höhe: wievielfach höher als die normalen Buchstaben im unteren Bildschirmteil.
2. Eingabe	Breite: wievielfach breiter als die normalen Buchstaben.
3. Eingabe	Codezahl für die Art der Schrift - z.B.:

Code	Schriftart
0	nach rechts, aufrecht (normal)
2	nach links, kopfstehend
4	nach links, aufrecht
5	nach oben, aufrecht
6	nach rechts, kopfstehend
7	nach unten, aufrecht
usw.	

4. Eingabe	Das zu "malende" Wort, oder der zu malende Satz.

Sie sollten etwas mit diesem Befehl und den 32 verschiedenen Codezahlen experimentieren. So lernen Sie schnell die vielfältigen Möglichkeiten.

Zusammenfassend kann man sagen, daß der Befehl MALE Wörter oder Sätze in vier Richtungen (mit jeder beliebigen Orientierung der einzelnen Buchstaben) und in beliebiger Größe schreiben kann.

3.6 Die Befehle SKALA und IGELGROESSE

Mit dem Befehl

- SKALA

können Sie das Verhältnis von X- und Y-Koordinaten für Ihren Bildschirm genau einstellen. Die Eingabe für diesen Befehl bestimmt den Korrekturfaktor. Dieser Faktor liegt im allgemeinen zwischen 1 und 1.5. Je größer Sie diesen Faktor bestimmen, um so "schmaler" werden die Zeichnungen. Damit verändert sich auch der Bereich der zugelassenen Werte für die X-Koordinate. Die Lage der Y-Achse wird nicht verändert.

Mit dem Befehl

- IGELGROESSE (kurz: IGR)

kann die Größe des Igels verändert werden (Voreinstellung: 6).
Die neue Größe des Igels wird durch eine als Eingabe gegebene Zahl bestimmt - es sind dabei Werte von 0.1 bis 127 zugelassen.
Ein Beispiel:

?STARTIGEL
?IGELGROESSE 12
?...

Der Igel ist erheblich größer geworden.

?IGR 3
?...

Der Igel ist sehr klein geworden.

?STARTIGEL
?...

Durch Aufruf des Befehl STARTIGEL bekommt der Igel auch wieder seine gewohnte Größe von 6.

Durch diesen Befehl IGELGROESSE lassen sich sehr schöne Effekte mit dem Igel erzielen.

4 Namen

Wir haben schon in verschiedenen Zusammenhängen von Namen gesprochen. Es gibt zum einen die Namen der Prozeduren und zum anderen die Namen der Eingaben dieser Prozeduren. Grundsätzlich müssen alle Namen in LOGO - ob die Namen der Prozeduren, die Namen von deren Eingaben oder auch die Dateinamen (s. Kap 5) - Wörter sein, die mit einem großen Buchstaben beginnen.

Es gibt nun noch eine weitere Möglichkeit, Namen zu definieren. Es ist in LOGO möglich, einem beliebigen Namen einen beliebigen Wert zuzuordnen. Dieser Wert kann eine Zahl, ein Wort oder eine Liste sein. Die Zuordnung wird durch den Befehl

- SETZE

erreicht.

```
?SETZE "KONST 21
?...
```

Es wird dem Namen KONST der Wert 21 zugeordnet.
Die Werte dieser Namen können genauso wie die Werte der Eingaben von Prozeduren durch einen Doppelpunkt vor dem Namen abgerufen werden.

Achtung :
Achten Sie darauf, keine Leerstelle zwischen dem Doppelpunkt und den Namen zu schreiben!

```
?DZ :KONST
21
?...
```

Der Doppelpunkt ist dabei lediglich eine Abkürzung und gehört eigentlich nicht mit zum Namen. Richtig müßte der Wert eines Namen durch den Befehl

- WERT

abgerufen werden. Dies gilt genauso für die Namen der Prozedureingaben.

```
?DZ WERT "KONST
21
?...
```

Der Name für den Befehl SETZE und den Befehl WERT muß durchaus nicht als Zeichenkette eingegeben werden. Er kann auch Ergebnis eines beliebigen LOGO-Grundbefehls oder einer Prozedur sein.

Im folgenden Beispiel wird der Befehl WORT benutzt. Dieser Befehl macht aus allen Buchstaben von zwei Eingaben ein neues Wort!

```
?SETZE (WORT "ZA "HL) 77                ; (WORT "ZA "HL) = ZAHL
?DZ WERT (WORT "Z "AHL)
77
?DZ :ZAHL
77
?...
```

Bei Verwendung des Doppelpunktes vor dem Namen als Abkürzung ist dies - leicht einzusehenderweise - nicht möglich. Deshalb ist der Befehl WERT gegenüber dem Doppelpunkt weitaus vielseitiger.

Anmerkung:
Obwohl der Doppelpunkt eine wichtige Rolle beim Aufruf von Namen spielt, so ist er bei der Anwendung von SETZE und WERT unbedeutend; d.h., er d a r f auch hier benutzt werden. Es kann manchmal von Vorteil sein, den Doppelpunkt auch bei SETZE und WERT mitzuschreiben. Dies kann mitunter eine Prozedur schneller machen.

4.1 Globale und lokale Namen

In allen höheren Programmiersprachen wird zwischen globalen und lokalen Variablen unterschieden. In LOGO bezeichnen wir die Variablen als Namen. Wir wollen von jetzt an die Namen von Prozeduren nicht mehr als "Namen" bezeichnen, sondern nur noch die Variablennamen.
Auch in LOGO unterscheiden wir zwischen globalen und lokalen Namen. Als globale Namen bezeichnen wir die Namen, die überall (global) bekannt sind - also von keiner Prozedur abhängig sind. Lokale Namen sind die, die nur innerhalb einer Prozedur - also eines bestimmten (lokalen) Bereichs - bekannt sind. Sowohl bei der Abfrage als auch bei dem Zuordnen haben die lokalen Namen Vorang. So kann es passieren, daß ein lokaler Name einen gobalen Namen verschattet.

```
PR DRUCKEZ :ZAHL
 DZ :ZAHL
ENDE
```

```
?SETZE "ZAHL 77
?DZ :ZAHL
77
?DRUCKZ 68
68
?...
```

Hier nun kommt es zu dem angesprochenen Konflikt zwischen einem globalen und einem lokalen Namen. Beide heißen "ZAHL". Da innerhalb der Prozedur DRUCKEZ der globale Name ZAHL durch den lokalen Namen ZAHL überdeckt wird, kann innerhalb dieser Prozedur nicht mehr auf den globalen Namen zurückgegriffen werden. Sowie die Prozedur dann beendet ist, ist wieder nur noch der globale Name ZAHL bekannt.

Sowohl die Werte von globalen als auch die von lokalen Namen können geändert werden. Hierzu braucht dem Namen nur der neue Wert mit SETZE zugeordnet zu werden. Es wird dann jeweils der Wert geändert, der auch durch WERT oder den Doppelpunkt abrufbar ist.

```
PR AENDERE :Y
 SETZE "Y (:Y * 2)
 DZ :Y
ENDE
```

```
?SETZE "Y 5
?AENDERE 15
30
?DZ :Y
5
?...
```

In dieser Prozedur AENDERE wird der Wert von dem lokalen Namen "Y" verändert. Dabei bleibt aber der Wert des globalen Namens "Y" erhalten. Aber auch dieser globale Name kann geändert werden:

```
?SETZE "Y :Y * 2
?DZ :Y
10
?...
```

Die Benutzung vieler solcher globaler Namen macht das Programmieren zwar meistens erst etwas einfacher, erhöht aber die Unübersichtlichkeit eines Programms ganz beträchtlich. Besonders, wenn an mehreren Stellen im Programm die Werte der globalen Namen geändert werden, entstehen sehr leicht Fehler, die dann sehr schwer zu finden sind. Im begrenzten Rahmen sind globale Namen aber eine wichtige Hilfe beim Programmieren.

Es gibt auch eine logische Frage danach, ob ein Name vorliegt.

```
PR VERSUCH :NUMMER
 DZ NAME? "NUMMER
ENDE
```

```
?SETZE "ZAHL 3
?NAME? "ZAHL
 Ergebnis = WAHR
?NAME? 23
 Ergebnis = FALSCH
?NAME? "NUMMER
 Ergebnis = FALSCH
?VERSUCH 0
WAHR
?...
```

Die Frage NAME? liefert als Ergebnis immer dann WAHR, wenn es sich bei der Eingabe um einen globalen oder lokalen Namen handelt.

5 Arbeitsspeicher und Datenverwaltung

5.1 Auslistungen von Prozeduren und Namen

Der Grundbefehl für das Auslisten von Daten lautet

- ZEIGE (kurz : ZG)

Dieser Befehl wird in Verbindung mit dem Namen einer Prozedur eingegeben. Dann wird diese Prozedur auf dem Bildschirm ausgedruckt.

```
?ZEIGE "QUADRAT
PR QUADRAT
 WH 4 [VW 50 RE 90]
ENDE
?...
```

Hier wurde die zuvor definierte Prozedur QUADRAT ausgedruckt.

Anmerkung : Der Name der PROZEDUR muß auch hier wie ein Wort mit Anführungszeichen eingegeben werden.

ZEIGE kann aber auch im Zusammenhang mit den LOGO-Grundwörtern:

- ALLES (kurz: AL),
- NAMEN (kurz: NA),
- PROZEDUREN (kurz: PR),
- TITEL oder
- EIGENSCHAFTEN (kurz: EIGS)

verwendet werden. Weil dies LOGO-Grundwörter sind, dürfen diese nicht mit Anführungszeichen eingegeben werden.

- ZEIGE ALLES (ZG AL)

Es wird der ganze Arbeitsspeicher (nicht die Eigenschaften) mit allen Prozeduren und Namen ausgedruckt.

```
?ZEIGE NAMEN              (ZG NA)
:ZAHL = 2
:LAENGE = 25
 ...
```

Es werden nur alle Namen mit ihren Werten ausgedruckt.

```
?ZEIGE PROZEDUREN          (ZG PR)
PR QUADRAT
 ...
ENDE
PR ...
```

Es werden nur alle Prozeduren ausgedruckt.

```
?ZEIGE TITEL               (ZT)
PR QUADRAT
PR RECHTECK
PR ZAEHLE
 ...
```

Da dieser Befehl besonders häufig benutzt wird, hat er sein eigenes Kürzel. Es werden die Namen aller Prozeduren ausgedruckt.

Der Befehl ZEIGE EIGENSCHAFTEN wird erst im Kapitel 9 behandelt.

Bitte vergessen Sie nie die Leertaste zwischen den Wörtern ZEIGE, ALLES...!

Sie können die Auslistungen mit "Ctrl-S" anhalten und dann mit jeder beliebigen Taste wieder fortsetzen. Durch "Ctrl-C" wird die Auslistung abgebrochen. (Siehe auch Kapitel 1.10)

5.2 Löschen von nicht mehr gebrauchten Daten

Sie können sich auch überflüssiger Prozeduren und Namen entledigen, wenn Sie den Speicherplatz für neue Daten benötigen. Der Grundbefehl dazu lautet:

- VERGISS (kurz: VG)

VERGISS wird entweder mit dem Namen einer Prozedur oder mit einem der schon oben genannten Wörter ALLES, NAMEN, PROZEDUREN oder EIGENSCHAFTEN benutzt.

VERGISS "QUADRAT

löscht die Prozedur QUADRAT aus dem Arbeitsspeicher.

- VERGISS ALLES (VG AL)

löscht den gesamten Arbeitsspeicher.

- VERGISS NAMEN (VG NA)

löscht alle Namen und deren Werte aus dem Arbeitsspeicher.

- VERGISS PROZEDUREN (VG PR)

löscht alle Prozeduren.

- VERGISS EIGENSCHAFTEN (VG EIGS)

löscht alle Eigenschaften, die im Arbeitsspeicher stehen.

Die Daten werden allerdings nicht wirklich gelöscht. LOGO vergißt lediglich deren Existenz. Erst bei Aufruf der nächsten "Garbage-collection" werden die Daten dann endgültig gelöscht.

5.3 Die Garbage-collection

(Garbage-collection = engl.: Müllsammlung)

Da AIP - LOGO ausschließlich mit Listen arbeitet und bei der Verarbeitung von Prozeduren oder Eingaben immer eine Menge nicht mehr gebrauchter Listen oder Listenteile entstehen, müssen diese von Zeit zu Zeit - wenn der Arbeitsspeicher zu Ende geht - gelöscht werden, um Platz für neue Daten zu schaffen. Diese Garbage-collection wird von AIP - LOGO automatisch aufgerufen, wenn der Speicher voll ist. Da diese Garbage-collection max. eine Sekunde dauert, werden Sie dies kaum bemerken.

Erst, wenn trotz der GC der Speicher immer noch zu voll ist meldet LOGO:

*** Speicher ist voll !

Dies tritt eigentlich nur während der Ausführung von großen, komplexen, datenreichen Programmen auf. Sie haben dann meist wieder etwas Platz zur Verfügung, wenn die Prozedur abgebrochen wurde, da dann alle lokalen Namen wieder gelöscht sind.

Die Garbage-collection kann auch durch den Befehl

- GCOLL

"von Hand" erzwungen werden. Dies ist ratsam, wenn Sie vermeiden wollen, daß die GC an anderer Stelle von AIP - LOGO aufgerufen wird und dort evt. einen Programmteil zeitlich verzögert.

5.4 Dateien

Die sich im Arbeitsbereich des Rechners befindenden Daten gehen leider nach dem Ausschalten des Gerätes verloren. Häufig wollen wir aber diese Daten, oder zumindest einen Teil davon, erhalten. Wir haben nun die Möglichkeit, die Daten - also Prozeduren und Namen - in einer Datei auf Cassette oder Diskette zu bewahren. Dabei werden nur globale und keine lokalen Variablen bewahrt.

ANMERKUNG :
Wenn Sie eine Datei schreiben oder lesen, und Sie befinden sich im Bild-Modus, genauer im TEILBILD-Modus, so schaltet AIP-LOGO während des Schreib- oder Lesevorgangs in den VOLLBILD-Modus über. Das hat folgenden Vorteil: Da AIP - LOGO für das Schreiben und Lesen von Dateien einen relativ großen Buffer benötigt, wird hierzu der Bildschirmbuffer - der sonst nur für das Teilbild benutzt wird - von den Bilddaten freigemacht. Dann kann dieser Buffer hier für das Lesen oder Schreiben genutzt werden.
Dies ist ebenso bei Bilddateien (s. Kap. 5.5).

5.4.1 Dateien bewahren

Der Befehl, etwas auf Cassette/Diskette zu schreiben, lautet

- BEWAHRE (Abkürzung : BW).

BEWAHRE kann ebenfalls in Verbindung mit ALLES, NAMEN, PROZEDUREN und EIGENSCHAFTEN benutzt werden.

```
?BEWAHRE ALLES "DATNAME          (BW AL ...)
 ...
 Bewahre DATEI: DATNAME
 ...
```

So werden alle Namen, Prozeduren und Eigenschaften des Arbeitsspeichers bewahrt. Dabei muß immer ein Dateiname mitangegeben werden. Diese Eingabe muß ein Wort sein (erstes Zeichen ein großer Buchstabe). AIP - LOGO meldet nun, daß die Datei "DATNAME" bewahrt wird. Dabei wird grundsätzlich automatisch eine Garbage-collection durchgeführt.

Wenn während des Schreibvorgangs ein Fehler auftritt, meldet LOGO :

```
 Bewahren nicht ok !
```

Bei Diskette wird die Art des Fehlers mit angegeben. Z.B.:

- Disk ist schreibgeschützt !
- Disk ist voll !
- Dateiname ist schon bekannt !

usw.

Läuft alles glatt, so wird das Programm (die Zeile) weiter ausgeführt.

Der Dateiname darf bei Diskette höchstens acht Zeichen und bei Cassette höchstens sechtzehn Zeichen lang sein. Bei zu langen Dateinamen werden die "überstehenden" Buchstaben einfach abgeschnitten.

- BEWAHRE NAMEN (BW NA)

Es werden alle Namen und deren Werte auf Cassette/Diskette geschrieben.

- BEWAHRE PROZEDUREN (BW PR)

Es werden alle Prozeduren bewahrt.

- BEWAHRE EIGENSCHAFTEN (BW EIGS)

Es werden alle Eigenschaften bewahrt.

Der Befehl BEWAHRE kann auch ohne eines dieser Worte aufgerufen werden. In diesem Fall benötigt AIP - LOGO hinter dem Dateinamen noch zwei weitere Eingaben. Dieses müssen beides Listen sein. Die erste Liste enthält alle Namen der Prozeduren, die bewahrt werden sollen, die zweite Liste alle globalen Namen, die bewahrt werden sollen. Es müssen also alle Namen und Prozeduren, die bewahrt werden sollen, in diesen beiden Listen einzeln aufgeführt werden. Dies hat den Vorteil, daß wir auswählen können, welche Namen und Prozeduren bewahrt werden sollen, und welche nicht.

?BEWAHRE "DATNAME [QUADRAT RECHTECK ZAEHLE] [:ZAHL :LAENGE]
...

Hier werden die Prozeduren QUADRAT, RECHTECK und ZAEHLE und die Namen :ZAHL und :LAENGE auf Cassette/Diskette geschrieben.

5.4.2 Datei einladen

Der Befehl, eine Datei wieder von Cassette /Diskette einzulesen, ist:

- LADE.

Ebenso wie der Befehl LADE wird auch der Befehl LADE$ benutzt (siehe Kap. 10.5). Es werden durch den Befehl LADE$ aber Maschinenprogramme usw. eingeladen.

Dann wird die angegebene Datei auf der Cassette/Diskette gesucht. Wenn AIP - LOGO dann (nur bei Cassette) eine Datei gefunden hat, wird die Meldung

```
DATNAME1 gefunden
```

ausgegeben. Wenn die gesuchte Datei gefunden wurde, wird

```
DATNAME gufunden                    ; nur bei Cassette
Lade DATEI: DATNAME
...
```

ausgegeben, und die betreffende Datei wird eingeladen.

Nun werden alle Prozeduren, Namen und Eigenschaften aus der Datei in den Arbeitsspeicher übernommen. Dabei werden alte Prozeduren, Namen und Eigenschaften nur dann gelöscht, wenn eine neue Prozedur (Name, Eigenschaft) mit dem gleichen Prozedurnamen (der gleichen Bezeichnung für den Namen / die Eigenschaft) eingelesen wird. Alle anderen Daten bleiben erhalten.

Wenn beim Laden ein Fehler auftritt (z.B. die gesuchte Datei nicht gefunden wurde), meldet das System

```
Laden ist nicht ok !
```

Bei Diskette wird auch die Art des aufgetretenen Fehlers beschrieben. Z.B.:

```
Dateinamen nicht gefunden !          ; nur Diskette
Laden ist nicht ok !
```

Bei fehlerfreiem Lesen wird das Programm bzw. die Zeile weiter ausgeführt.

Beim Einladen einer Datei ist es unerheblich, ob diese Datei mit BEWAHRE ALLES, BEWAHRE NAMEN, BEWAHRE PROZEDUREN oder BEWAHRE EIGENSCHAFTEN erzeugt worden ist.

Nun noch eine Vereinfachung des Einladens von Cassette:

Wenn zu dem Befehl LADE (auch LADE$ und LADEBILD s.u.) kein Dateiname als Eingabe gegeben wird, sondern eine Zahl (z.B. 0), dann wird die erste auf der Cassette gefundene Datei geladen. So können wir auch eine Datei einladen, ohne ihren genauen Dateinamen zu kennen:

```
?LADE 0                              ; nur Cassette !!
 ...
 DATEINAME1 gefunden                     ; nur Cassette !
 Lade DATEINAME1
 ...
```

5.5 Bilddateien

Es kann auch das aktuelle - sich auf dem Bildschirm befindliche - Bild bewahrt werden (natürlich nur im BILD-Modus). Dazu wird der Befehl

- BEWAHREBILD (kurz: BWB)

mit dem Dateinamen als Eingabe benutzt.

```
?BEWAHREBILD "KREISE
 ...
 Bewahre BILD: KREISE
 ...
```

Nach fehlerfreiem Schreiben wird das Programm oder die Zeile weiter ausgeführt.

Anmerkung:
Da das gesamte Bild auf Ihrem Computer ziemlich speicheraufwendig ist, dauert das Schreiben und Lesen einer Bilddatei auf Cassette recht lange. Haben Sie bitte Geduld.

Mit dem Befehl

- LADEBILD (kurz: LDB),

mit dem Dateinamen als Eingabe, wird das Bild wieder eingelesen.

```
?LADEBILD "KREISE
 ...
 Lade BILD: KREISE
 ...
```

5.6 Weitere Diskettenbefehle

Wenn Sie eine Diskette haben, können Sie sich mit dem Befehl

- INHALT (kurz: IH)

den Inhalt (das Directory) der Diskette zeigen lassen.
Es werden alle Files auf der Diskette auf dem Bildschirm gelistet.

```
?INHALT
 AIP_LOGO.BIN : DATEI1  .LOG
 DATEI2  .LOG : MUSTER  .BI4
 ...
```

Bilddateien werden immer mit dem Zusatz .BIx ("x" steht für eine Rechnerabhängige Ziffer von 0 bis 9.) für "BIld" versehen.
Programm und Datenfiles haben immer den Zusatz LOG für "LOGO".

Sie können auf Diskette auch Dateien und Bilder wieder löschen.

Mit dem Befehlen

- VERGISSBILD (kurz: VGB)

und

- VERGISSDATEI (kurz: VGD)

können Bilder bzw. Dateien auf Diskette gelöscht werden. Zwei Beispiele:

```
?VERGISSBILD "MUSTER
?...
```

Es wird die Bilddatei mit dem NAMEN "MUSTER .BIx" auf der Diskette gelöscht.

```
?VERGISSDATEI "DATEI2
?...
```

Es wird die Datei "DATEI2 .LOG" auf der Diskette gelöscht.

Wenn Sie für Ihren Computer mehr als ein Diskettenlaufwerk haben, so können Sie mit dem Befehl

- FDD

(für Floppy Disk Drive) mit Eingabe einer Zahl zwischen 1 und 4 auf ein anderes Laufwerk umschalten:

```
?FDD 2
?...
```

Es wird das Laufwerk "B" (das Zweite) angemeldet.

6. Zahlen, Wörter, Listen

6.1 Zahlen und mathematische Operationen

Grundsätzliches über Zahlen in AIP - LOGO:

- Statt des Kommas in Realzahlen wird immer ein Punkt (Dezimalpunkt) eingegeben.
- Vor dem Dezimalpunkt führende Nullen können - aber brauchen nicht - miteingegeben werden. Ebenso ist es mit den Nullen als letzte Stellen hinter dem Komma.
- Negative Zahlen werden durch ein "-" vor der Zahl dargestellt. Zur Unterscheidung mit dem Rechensymbol ist es meistens notwendig, negative Zahlen in Klammern zu fassen. Z.B. (-2.5).
- Positive Zahlen brauchen n i c h t mit einem "+" eingegeben zu werden. Dies würde nur wieder zu Verwechslungen mit dem Rechensymbol führen.

In LOGO gibt es zwei verschiedene Typen von Zahlen.

- ganze Zahlen (Integerzahlen)
- Realzahlen (Zahlen mit Komma und/oder Exponent)

Es ist Ihnen bisher sicherlich nicht aufgefallen, daß LOGO mit zwei verschiedenen Zahlentypen arbeitet. Das liegt daran, daß LOGO diese Zahlentypen immer dann, wenn es nötig ist, ineinander umwandelt. Fehler treten erst auf, wenn dies nicht möglich ist:

```
?WIEDERHOLE 3.1 [DZ "LOGO]
 Ganze Zahl erwartet von : WIEDERHOLE
?...
```

Viele Befehle in LOGO brauchen ganze Zahlen als Eingaben. Wenn für den Befehl WIEDERHOLE eine Kommazahl eingegeben wird, meldet AIP - LOGO einen Fehler.

- INT
- RUNDE

Um eine Realzahl in eine ganze Zahl umzuwandeln, ist der Befehl INT zu verwenden. Er ermittelt die nächst kleinere ganze Zahl.
Der Befehl RUNDE dagegen ermittelt die der Eingabe näheste ganze Zahl. Dieser Vorgang wird als Runden einer Zahl bezeichnet.

```
?DZ INT 3.2
3
?DZ INT -3.1
-4
```

```
?DZ RUNDE 3.1
3
?DZ RUNDE 3.8
4
?DZ RUNDE 3.5
4
?...
```

Die Befehle INT und RUNDE melden dann einen Fehler, wenn die sich ergebende ganze Zahl außerhalb des zugelassenen Bereichs (s. Kap. 6.1.1) liegt.

Die Frage danach, ob eine Zahl vorliegt, heißt:

- ZAHL?

Sie liefert als Ergebnis WAHR, wenn die Eingabe eine Zahl ist. Diese Frage kann, wie die meisten Fragen in AIP - LOGO, mit mehreren Eingaben (in runde Klammern gefaßt) aufgerufen werden. Dann wird nur in dem Fall WAHR geliefert, wenn alle Eingaben Zahlen sind.

```
?ZAHL? 3.56
 Ergebnis = WAHR
?ZAHL? "BAUM
 Ergebnis = FALSCH
?(ZAHL? 3 4.1 "TORTE)
 Ergebnis = FALSCH
?...
```

6.1.1 Die ganzen Zahlen (Integerzahlen)

Die ganzen Zahlen werden von AIP - LOGO intern als 32 Bit abgelegt. Daraus ergibt sich die kleinste ganze Zahl:

```
 -2147483647                          ; - (2 hoch 31) + 1
```

Die größte darstellbare ganze Zahl ist:

```
 +2147483647                          ; + (2 hoch 31) - 1
```

Es gibt die Frage, ob eine ganze Zahl vorliegt:

- IZAHL?

Diese Frage kann auch wieder mit mehreren Eingaben (dann in runden Klammern) aufgerufen werden.

```
?IZAHL? 3
 Ergebnis = WAHR
?DZ (IZAHL? 3 0 5.1)
FALSCH
?...
```

VORSICHT:
Nicht alles, was wie eine ganze Zahl aussieht, ist auch eine. Es kann sich auch um eine Zahl handeln, die zwar dem Wert nach eine ganze Zahl ist, der Form (intern) nach aber eine Realzahl.

Dazu ein Beispiel:

```
?IZAHL? 1.5 * 2
 Ergebnis = FALSCH
?...
```

Das Ergebnis der Rechnung 1.5 * 2 ist zwar die ganze Zahl 3, intern aber die Realzahl 3! Dies ist aber auch nur für die Frage IZAHL? von Bedeutung.

Es gibt in AIP - LOGO zwei reine Integerfunktionen (Ganzzahloperationen) :

- DIV
- REST

```
?DIV 26 3
 Ergebnis = 8
?...
```

Der Befehl DIV berechnet den ganzzahligen Anteil von 26 / 3, ist also gleichbedeutend mit (INT 26 / 3).

```
?REST 26 3
 Ergebnis = 2
?...
```

Der Befehl REST berechnet den Rest, der bei der Rechnung 26 / 3 auftritt. Das ist gleichbedeutend mit (26 - 3 * DIV 26 3).
Allerdings wird bei diesen beiden Integerfunktionen nicht vorübergehend mit Realzahlen gerechnet. Das bedeutet, daß DIV und REST schneller sind.

Aber auch die mathematischen Operationen +,-,* (nicht: /,**) werden zwischen ganzen Zahlen durchgeführt. Vor allen anderen mathematischen Operationen muß AIP - LOGO ganze Zahlen erst in Realzahlen umformen.

6.1.2 Realzahlen

Die Realzahlen in AIP - LOGO können Werte zwischen +-0.271050544 mal 10 hoch -19 und +-0.922337203 mal 10 hoch +19. Nur wenn beide Eingaben WAHR sind, wird WAHR zurückgegeben, sonst FALSCH.

```
?DZ ALLE? ZAHL? 2 ZAHL? "ANJA
FALSCH
?DZ ALLE? ZAHL? 2 ZAHL? 0
WAHR
?...
```

Auch diese beiden logischen Fragen können in AIP - LOGO mit beliebig vielen Eingaben aufgerufen werden - in runden Klammern selbstverständlich. Es wird dann jeweils gefragt, ob eine der Eingaben WAHR ist, bzw., ob alle Eingaben WAHR sind. So ist es logisch, daß die Frage EINES? ohne eine Eingabe FALSCH ergeben muß, während die Frage AL * 10 hoch - 1

Negative Exponenten werden also nicht wie in z.B. BASIC durch ein Minuszeichen davor, sondern durch ein "N" statt dem "E" dargestellt. Sie müssen hier darauf achten, daß die ganze Realzahl in einem Wort eingegeben wird und nicht mit Leerstellen dazwischen.

Wenn Sie eine Realzahl falsch eingeben, meldet AIP - LOGO dies mit einer besonderen Fehlermeldung:

```
?DZ 3.3333E4 * 4545S3N1
 Zahl nicht in Ordnung : 4545S3N1
?...
```

AIP - LOGO erkennt, daß der Ausdruck 4545S3N1 eine Zahl sein soll, und zwar an dem ersten Zeichen, weil dies eine Ziffer bzw. ein Dezimalpunkt ist. Da diese Zahl aber nicht sinnvoll ist, meldet AIP - LOGO einen Fehler.

6.1.3 Operationen mit Realzahlen

Wie in jeder Programmiersprache mit Realzahlen bietet auch AIP - LOGO viele Befehle (Funktionen), die Realzahlen verarbeiten.
Alle Realzahlen verarbeitenden Funktionen werden jetzt mit je einem oder zwei Beispielen aufgeführt.

Die trigeometrischen Funktionen:

- SIN = sinus
- COS = cosinus
- TAN = tangens
- ARCTAN = arcus tangens

```
?SIN 30
 Ergebnis = 0.5
?DZ COS 90
0
?TAN 0
 Ergebnis = 0
?DZ ARCTAN 45
88.72697
?...
```

Die Winkel für diese Funktionen werden immer in Grad (degrees) angegeben bzw. berechnet (360 Grad sind ein voller Kreis!).

Die Exponentialfunktionen:

- LOG Logarythmus zur Basis 10.
- LN Logarythmus zur Basis e (=2.718281828).
- EXP e hoch x.

```
?DZ LOG 100
2
?DZ LN 100
4.6051702
?EXP 30.30
 Ergebnis = .14425232E14
?...
```

Die Quadratwurzel:

- WURZEL (kurz: QW)

```
?WURZEL 625
 Ergebnis = 25
?...
```

In LOGO ist auch ein Zufallszahlengenerator eingebaut. Mit dem Befehl

- ZUFALLSZAHL (kurz: ZZ)

und einer Zahl als Eingabe, wird eine zufällige ganze Zahl größer gleich Null und kleiner der eingegebenen Zahl als Ergebnis zurückgegeben.

```
?DZ ZUFALLSZAHL 4
2                                   ; möglich : 0,1,2,3
?DZ ZUFALLSZAHL 9
3                                   ; möglich : 0,1,2...7,8
?DZ ZUFALLSZAHL 9
6                                   ; möglich : 0,1,2...7,8
?...
```

Mit dem Befehl

- STARTEZUFALL (kurz: SZ)

(ohne Eingaben) werden die Bedingungen für ZUFALLSZAHL wieder zurückgesetzt, d.h., wenn die Zufallszahlen wieder in der gleichen Reihenfolge aufgerufen werden, erneut die gleichen Zahlen geliefert werden. Dies ist besonders sinnvoll, wenn Fehler in Programmen gefunden werden sollen, da so der Ablauf eines Programms wieder genau nachvollzogen werden kann.

6.2 Wörter

Wir haben die Wörter in LOGO ja schon kennengelernt. Ein LOGO-Wort kann dann auch als LOGO-Name verwendet werden, wenn es mit einem großen Buchstaben beginnt. Auch Zahlen können Wörter sein, denn ein Wort ist als eine Zeichenkette festgelegt.
Es gibt in AIP - LOGO nun einige Befehle, die sich auf Wörter beziehen. Wenn die Eingaben für solche Befehle Zahlen sind, so werden diese vorher von AIP - LOGO in eine Zeichenkette umgearbeitet.

Wortverarbeitende Befehle:

- ERSTES (kurz ER)

Dieser Befehl liefert den ersten Buchstaben des Wortes (also ein neues Wort mit nur diesem Buchstaben).

- OHNEERSTES (kurz OE)

Es wird das Wort ohne den ersten Buchstaben zurückgegeben.

- LETZTES (kurz LE)

Es wird der letzte Buchstabe des Wortes als neues Wort zurückgegeben.

- OHNELETZTES (kurz OL)

Es wird das Wort ohne den letzten Buchstaben zurückgegeben.

Beispiele:

```
?DZ ERSTES "BAUM
B
?DZ ER "LOGO
L
?DZ ERSTES 3.1415
3
?DZ LETZTES "LOGO
O
?DZ OHNEERSTES "BAUM
AUM
?DZ ERSTES OHNEERSTES "BAUM              ; = ERSTES "AUM    !
A
?DZ OHNELETZTES "LOGO
LOG
?...
```

Der Befehl

- UMKEHR

dreht ein Wort - Zeichen für Zeichen - um. Danach ist das erste Zeichen zum letzten, das zweite zum vorletzten usw. geworden.

```
?UMKEHR "TORTE
 Ergebnis = ETROT
?...
```

Der Befehl

- WORT

macht aus allen Buchstaben, die in zwei Eingaben vorkommen, ein neues Wort. Wenn eine der Eingaben eine Liste ist, so werden die Buchstaben aller in der Liste erscheinenden Wörter genommen (keine Klammern!).

Es ist auch möglich, den Befehl WORT mit mehr oder weniger als zwei Eingaben (in runde Klammern gefaßt!) aufzurufen.

```
?WORT "BA "UM
 Ergebnis = BAUM
?DZ WORT [DAS IST EIN] "BAUM
DASISTEINBAUM
?DZ (WORT "DAS "IST "EINE 1.1)
DASISTEINE1.1
?...
```

Die Frage

- WORT?

prüft, ob ein Wort vorliegt. Dabei sind ganze Zahlen und auch Realzahlen k e i n e Wörter (aber auch nur für diese Frage!). Auch diese Frage kann mehrere Eingaben (in runden Klammern) bekommen.

```
?DZ WORT? "BAUM
WAHR
?DZ (WORT? 23 "WIESE)
FALSCH
?...
```

6.3 Die Listen

Wir haben in LOGO schon Listen kennengelernt. Wenn wir z.B. einen Satz ausdrucken wollen, müssen wir für DRUCKEZEILE eine Liste mit den Wörtern des Satzes eingeben. Dabei muß die Liste durch eckige Klammern zusammengehalten werden.

```
?DRUCKEZEILE [DIES IST EIN SATZ]
DIES IST EIN SATZ
?...
```

Die Liste besteht hier aus vier Elementen. Alle Elemente sind Wörter. Es können aber auch Zahlen oder auch Listen Elemente von Listen sein.
Dabei ist aber etwas zu beachten: NUR die äußersten Klammern dürfen eckig sein, alle "Unterlisten" (Listen, die Elemente der Liste sind) werden durch runde Klammern zusammengehalten. Eigentlich gibt es in AIP - LOGO überhaupt nur die runden Klammern. Ein Liste sieht so aus:

```
(DIES IST (FAST) EINE LISTE)
```

Hier könnte AIP - LOGO aber nicht mehr zwischen Listen und Befehlen (da diese ja auch in runden Klammern aufgerufen werden können) unterscheiden. Um eine Unterscheidung zu ermöglichen, machen wir nun mit den Listen das gleiche wie mit den Wörtern: Wir setzen ein Anführungszeichen davor.

```
"(DIES IST EINE (ECHTE) LISTE)
```

So können Sie die Liste nun eingeben. Die eckigen Klammern bedeuten auch nichts anderes. Sie sind nur eine kürzere Schreibweise für die runden Klammern mit einem Anführungszeichen. Intern besteht für AIP - LOGO kein Unterschied. Sie werden auch bemerken, daß, wenn Sie in einer Prozedur "(...) schreiben und sich die Prozedur dann zeigen lassen, AIP - LOGO dafür eine eckige Klammer [...] ausgibt. Dadurch werden Programme erheblich besser lesbar!

Beispiele für Listen:

```
?"(DIESER BAUM IST 10 JAHRE ALT (DAS IST NOCH SEHR JUNG))
 Ergebnis = (DIESER BAUM IST 10 JAHRE ALT (DAS IST NOCH SEHR JUNG))
?DZ [EINE LISTE]
EINE LISTE
?DZ "(EINE LISTE)
EINE LISTE
?DZ [34.6 (A (B () ((C)) 1) (1) (((3))))]
34.6 (A (B () ((C)) 1) (1) (((3))))
?...
```

Wie wir sehen, können Listen beliebig geschachtelt werden.

Aber eines fällt uns sofort auf: Der Befehl DRUCKEZEILE (DZ) läßt beim Drucken einer Liste immer die äußersten Klammern weg. Dies ist sinnvoll, da ja häufig Sätze ausgedruckt werden sollen.
Nicht weggelassen werden diese äußersten Klammern jedoch, wenn AIP - LOGO das Ergebnis als "Ergebnis =" ausgibt:

```
?[1 2 3 4 5 (6) 7 8 9 0]
 Ergebnis = (1 2 3 4 5 (6) 7 8 9 0)          ; äußerste Klammern bleiben
?...                                         ; erhalten !
```

6.3.1 Die leere Liste

Eine besondere Form der Liste, die leere Liste, ist ein Liste o h n e Element. Sie taucht oft als Ergebnis von listenverarbeitenden Operationen auf. Sie hat aber in LOGO nur eine geringe Bedeutung (z.B. verglichen mit der Programmiersprache LISP).
Die leere Liste wird von dem Befehl DRUCKEZEILE als leere Zeile ausgegeben. Wir können auch direkt eine leere Liste eingeben.

```
?[]
 Ergebnis = ()
?DZ []
                                    ; eine leere Zeile wird ausgegeben!
?...
```

6.3.2 Die Grundbefehle der Listenverarbeitung

Für die Listen gibt es einen ähnlichen (erweiterten) Befehlssatz wie für die Wörter.

Die folgenden Befehle können auch für Listen, wie es Ihnen für die Worte bekannt ist, benutzt werden.

- ERSTES (kurz: ER)
- OHNEERSTES (kurz: OE)
- LETZTES (kurz: LZ)
- OHNELETZTES (kurz: OL)

Einige Beispiele:

```
?DZ ERSTES [DIES IST EINE LISTE]
DIES
?DZ OHNEERSTES [DIES IST EINE LISTE]
IST EINE LISTE
?DZ LETZTES [DIES IST EINE LISTE]
LISTE
?DZ OHNELETZTES [DIES IST EINE LISTE]
DIES IST EINE
?...
```

Mit den Befehlen

- MITERSTEM (kurz: ME)
- MITLETZTEM (kurz: ML)

können Listen auch um ein Element vorne oder hinten erweitert werden.

```
?MITERSTEM "BAUM [DIES IST EINE LISTE]
 Ergebnis = (BAUM DIES IST EINE LISTE)
?DZ MITERSTEM "BAUM "WIESE
BAUM WIESE
?DZ MITLETZTEM "BAUM [DIES IST EINE LISTE]
DIES IST EINE LISTE BAUM
?DZ MITLETZTEM "BAUM "WIESE
WIESE BAUM
?...
```

Der Befehl MITERSTEM fügt das als erste Eingabe gegebene Objekt vorn an die als zweite Eingabe gegebene Liste an. Wenn als zweite Eingabe keine Liste gegeben wird, so wird aus dem gegebenen Objekt eine Liste gemacht. Der Befehl MITLETZTEM fügt die erste Eingabe hinten an die Liste (zweite Eingabe) an, ist ansonsten aber gleich.

Weitere listenverarbeitende Befehle:

- LISTE
- SATZ

Diese beiden Befehle unterscheiden sich nur in einer kleinen Feinheit:

```
?DZ LISTE "BAUM "WIESE
BAUM WIESE
?DZ LISTE [BAUM] [WIESE]
(BAUM) (WIESE)
?DZ SATZ "BAUM "WIESE
BAUM WIESE
?DZ SATZ [BAUM] [WIESE]
BAUM WIESE
?DZ SATZ [BAUM WIESE (TORTE)] "KAEFER
BAUM WIESE (TORTE) KAEFER
?...
```

Die Befehle SATZ und LISTE unterscheiden sich insofern, als LISTE immer eine neue Liste schafft, die die Eingaben als Elemente enthält, egal, ob es sich um Wörter, Zahlen oder Listen handelt. Bei Aufruf des Befehls SATZ versucht AIP - LOGO aus den Eingaben einen Satz zu bilden. Es entsteht dabei eine Liste, die die Elemente oberster Ebene von den Eingaben enthält (wenn es sich bei einer Eingabe um keine Liste handelt, wird die Eingabe selbst übernommen). So können mehrere Sätze (Satzteile) zu einem zusammengefaßt werden.

Beide Befehle SATZ und LISTE können (in runden Klammern aufgerufen) eine beliebige Anzahl von Eingaben verarbeiten:

```
?DZ (SATZ [DIES IST EIN SATZ] [DIES IST EIN] "ZWEITER)
DIES IST EIN SATZ DIES IST EIN ZWEITER
?(LISTE [DIES IST EIN SATZ] 1 "WORT [BAUM])
 Ergebnis = ((DIES IST EIN SATZ) 1 WORT (BAUM))
?...
```

Die Frage

- LISTE?

überprüft, ob eine Liste (mit mind. einem Element) vorliegt. Auch diese Frage kann in Verbindung mit runden Klammern mehrere Eingaben bekommen.

```
?LISTE? [DIES IST EINE LISTE]
 Ergebnis = WAHR
?DZ (LISTE? [DIES IST EINE LISTE] [])              ; leere Liste !
 Ergebnis = FALSCH
?...
```

6.3.3 Effizienz der Listenverarbeitung

Aufgrund der internen Struktur von Listen ist es für AIP - LOGO nur möglich, auf das jeweils erste Element einer Liste bzw. die Restliste (Liste ohne das erste Element) zuzugreifen (siehe auch Kap 10). Dadurch sind die Befehle ERSTES und OHNEERSTES sowie MITERSTEM extrem schnell (sie werden auch für die interne Datenverwaltung benutzt). Wenn nun aber der Befehl LETZTES, OHNELETZTES oder MITLETZTEM aufgerufen wird, muß AIP - LOGO die ganze Liste abarbeiten, bis es beim letzten Element angekommen ist, und dann (bei MITLETZTEM und OHNELETZTES) noch eine völlig neue Liste konstruieren. Dies bedeutet, daß der Zeitaufwand für die Befehle ERSTES usw. immer gleich ist, während er für die Befehle LETZTES usw. mit der Länge der Liste zunimmt. Dies kostet bei langen Listen viel Zeit.

Dazu folgende Beispiele:

```
?ERSTES [1 2 3 4 5 6 7 8 9 0]                      ; Zeile 1
 Ergebnis = 1
?LETZTES [1 2 3 4 5 6 7 8 9 0]                     ; Zeile 2
 Ergebnis = 0
?OHNEERSTES [1 2 3 4 5 6 7 8 9 0]                  ; Zeile 3
 Ergebnis = (2 3 4 5 6 7 8 9 0)
?OHNELETZTES [1 2 3 4 5 6 7 8 9 0]                 ; Zeile 4
 Ergebnis = (1 2 3 4 5 6 7 8 9)
?...
```

Die interne Ausführung des Befehls ERSTES (Zeile 1) benötigt ca. 1/100000 s (eine einhundertausendstel Sekunde). Die interne Ausführung von LETZTES (Zeile 2) benötigt schon etwa 1/20000 s (eine zwanzigtausendstel Sekunde).

Bei den Befehlen in Zeile 3 und 4 ist der Unterschied noch deutlicher:
Der Befehl OHNEERSTES in Zeile 3 benötigt, ebenso wie ERSTES in Zeile 1, intern etwa 1/100000 s. Der Befehl OHNELETZTES in Zeile 4 braucht dagegen schon etwa 1/2000 s (eine zweitausendstel Sekunde). Natürlich wird hier die Garbage-collection (s. Kapitel 5.3) noch nicht einmal berücksichtigt.

Man muß allerdings erwähnen, daß in diesen Beispielen der Zeitunterschied der internen Ausführung im Verhältnis zum Zeitaufwand des Einlesens der eingegebenen Zeile, sowie des Ausdruckens des Ergebnisses, noch so gering ist, daß es nach außen hin nicht zu merken ist.
Dieser Zeitunterschied kommt erst bei länger laufenden Prozeduren zum Tragen.

Näheres über die interne Struktur der Listen erfahren Sie in Kapitel 10. Sie sollten aber immer versuchen - soweit dieses ohne Mehraufwand möglich ist -, vorzugsweise auf die Befehle ERSTES, OHNEERSTES und MITERSTEM zuzugreifen!

6.4 Anwendungen von listenverarbeitenden Befehlen

Es werden nun verschiedene LOGO-Prozeduren vorgestellt, die alle auch als LOGO-Grundbefehl vorliegen. In diesen Prozeduren ist bewußt auf die Befehle MITLETZTEM, LETZTES, OHNELETZTES verzichtet worden, da diese, wie in Kapitel 6.3.3 beschrieben, uneffizient (langsam) sind.

Als LOGO - Einsteiger sollten Sie nicht gleich verzweifeln, wenn Sie die im folgenden aufgeführten Prozeduren nicht verstehen. Übergehen Sie diese dann einfach. Sie können natürlich die von AIP - LOGO zur Verfügung gestellten Befehle trotzdem benutzen!

6.4.1 Die Prozedur KLAMMERLISTE

Wir wollen eine Prozedur schaffen, die alle Elemente einer Liste klammert. Wenn keine Liste eingegeben wird, soll die Prozedur die Eingabe selbst als Liste zurückgeben.
Dazu müssen wir die Liste in ihre Elemente aufteilen, diese klammern und dann wieder zu einer Liste zusammenfügen. Dieses Problem läßt sich am bestem mit einer rekursiven Prozedur lösen.

```
PR KLAMMERLISTE :L
 TESTE LISTE? :L
 WENNFALSCH RUECKGABE (LISTE :L)
 RUECKGABE KLAMMERLI :L
ENDE
PR KLAMMERLI :L
 TESTE :L = []
 WW RUECKGABE :L
 RUECKGABE MITERSTEM (LISTE ER :L) (KLAMMERLI OE :L)
ENDE
```

```
?KLAMMERLISTE "BAUM
 Ergebnis = (BAUM)
?KLAMMERLISTE [BAUM WIESE KAEFER]
 Ergebnis = ((BAUM) (WIESE) (KAEFER))
?...
```

Wir haben das Problem mit zwei Prozeduren gelöst. Die eine Prozedur ist eine Hilfsprozedur (KLAMMERLI), die nach außen hin unsichtbar bleibt. Aber gerade in ihr wird die eigentliche Operation "Klammern" durchgeführt. Die Prozedur KLAMMERLISTE übernimmt nur den Test, ob wirklich - wie gefordert - eine Liste eingegeben wurde. Wenn ja, dann wird nur die Prozedur KLAMMERLI aufgerufen. Wenn aber nicht, so wird das Wort oder die Zahl selbst geklammert zurückgegeben.

Die Prozedur KLAMMERLI ist rekursiv. Die Abbruchbedingung ist, daß die übergebene Liste leer ist. Erst dann sind alle Elemente abgearbeitet worden. Die eigentliche Rekursion findet in der dritten Zeile statt. Es wird das erste Element der Liste geklammert (LISTE ER :L) und vorne an das Ergebnis von (KLAMMERLI OE :L) angefügt. Bevor LOGO es anfügen kann, muß LOGO die Prozedur KLAMMERLI erst auf die Liste ohne das erste Element anwenden. So vertieft sich die Prozedur so lange, bis die Liste leer ist. Erst dann wird die geklammerte Liste erstellt, indem ein Element nach dem anderen (jetzt jeweils schon geklammert) wieder vor die Liste gesetzt wird (mit dem Befehl MITERSTEM).

Diese Prozedur KLAMMERLISTE heißt in AIP - LOGO

- KLAMMER

```
?KLAMMER [1 2 3 (4) ((5) (6))]
 Ergebnis = ((1) (2) (3) ((4)) (((5) (6))))
?KLAMMER 3
 Ergebnis = (3)
?...
```

Alle Elemente der Liste wurden geklammert. Wenn als Eingabe keine Liste gegeben wird, so wird das Wort oder die Zahl als Liste zurückgegeben.

6.4.2 Die Prozedur LISTENLAENGE

Es soll jetzt eine Prozedur geschaffen werden, die die Anzahl von Elementen in einer Liste zählt. Wenn keine Liste eingegeben wird, so soll null als Ergebnis geliefert werden. Auch hier bietet sich die rekursive Form an.

```
PR LISTENLAENGE :L
 TESTE LISTE? :L
 WENNFALSCH RUCKGABE 0
 RG (1 + LISTENLAENGE OE :L)
ENDE
```

```
?DZ LISTENLAENGE "BAUM
0
?DZ LISTENLAENGE [BAUM WIESE KAEFER SEEROSE]
4
?...
```

Diese Prozedur zählt die Elemente der Liste, indem es die Elemente der Liste ohne das erste Element + 1 zurückgibt. Hat die Liste schließlich kein Element mehr, so wird null zurückgegeben (LISTE? [] ergibt FALSCH!). Diese Prozedur heißt in AIP - LOGO

- LAENGE.

```
?LAENGE [1 2 3 4 5 6]
 Ergebnis = 6
?LAENGE "BAUM
 Ergebnis = 0
?...
```

Es wird die Länge einer Liste ermittelt. Wenn keine Liste - oder eine leere Liste - vorliegt, wird null zurückgegeben.
Wir wollen diese Prozedur LAENGE gleich verwenden.

6.4.3 Die Prozedur PICKEELEMENT

Eine Prozedur soll geschrieben werden, die ein beliebiges Element, dessen Nummer wir eingeben, aus einer Liste heraussucht. Die Prozedur soll dabei einen Fehler melden, wenn die eingegebene Zahl so groß ist, daß die Liste nicht genügend Elemente enthält. Auch diese Aufgabe läßt sich einfach mit einer rekursiven Prozedur lösen.

Wir benutzen wieder zwei Prozeduren:

```
PR PICKEELEMENT :NR :L
 TESTE :NR <= LAENGE :L
 WF DZ [NUMMER IST ZU GROSS!] RK                    ; Fehlermeldung !
 RG PICKEEL :NR :L
ENDE
PR PICKEEL :NR :L
 TESTE :NR = 1
 WW RG ERSTES :L
 RG PICKEEL (:NR - 1) (OE :L)
ENDE

?PICKEELEMENT 4 [A B C]
NUMMER IST ZU GROSS!
?PICKEELEMENT 4 [A B C D E F]
 Ergebnis = D
?...
```

Die Prozedure PICKEELEMENT prüft erst, ob es ein 4. Element in der Liste überhaupt gibt. Wenn ja, wird die Prozedur PICKEEL aufgerufen. Hier findet wieder die Rekursion statt. PICKEEL arbeitet die Liste solange runter, bis :NR = 1 ist, und gibt dann das erste Element zurück.

In AIP - LOGO heißt dieser Befehl

- PICKE.

```
?PICKE 3 [A B C D]
 Ergebnis = C
?PICKE 12 [1 2 3 4]
 Ergebnis = ()
?...
```

Der Befehl PICKE sucht das n-te Element aus einer Liste. Wenn die Zahl (erste Eingabe) zu groß ist, oder wenn keine Liste (zweite Eingabe) gegeben wird, so wird die leere Liste als Ergebnis zurückgegeben.

6.4.4 Die Prozedur KEHRUM

Diese Prozedur KEHRUM soll eine Liste umdrehen. Das erste Element soll letztes werden, das zweite vorletztes usw.. Dabei ergibt sich das Problem, daß dies auf die bekannte Weise mit Rekursion nicht so ohne weiteres zu lösen ist. Um die Reihenfolge der Elemente umkehren zu können, müssen wir uns eine zweite Liste konstruieren, die zu Beginn der Prozedur leer ist und auf die ein Element nach dem anderen aufgesetzt wird (mit MITERSTEM). Wir wollen diese Liste als AKKU bezeichnen. Währenddessen wird die Originalliste Element für Element abgearbeitet. Das ist z.B. so möglich:

```
PR KEHRUM :L
 TESTE LISTE? :L
 WF DZ "FEHLER! RK                    ; Fehlermeldung !
 RG KEHRUM1 :L []
ENDE
PR KEHRUM1 :L :AKKU
 WENN :L = [] DANN RUECKGABE :AKKU
 RG KEHRUM1 (OE :L) (ME (ERSTES :L) :AKKU)
ENDE

?KEHRUM "BAUM
FEHLER!
?KEHRUM [BAUM WIESE KAEFER SEEROSE TONNE]
 Ergebnis = (TONNE SEEROSE KAEFER WIESE BAUM)
?...
```

Der lokale Name AKKU ist von außen nicht sichtbar. Er dient nur als Zwischenspeicher.

Auch die Prozedur KEHRUM ist AIP - LOGO schon bekannt. Sie heißt

- UMKEHR.

Wir kennen diesen Befehl schon von der Wortverarbeitung.

```
?UMKEHR [1 2 3 4 5]
 Ergebnis = (5 4 3 2 1)
?UMKEHR "BAUM
 Ergebnis = MUAB
?...
```

6.4.5 Die Prozedur GLAETTELISTE

Der Befehl

- GLAETTE

entfernt alle Klammern aus einer Liste. Es entsteht also eine Liste, die als Elemente nur die Wörter und die Zahlen beinhaltet, die in der Ausgangsliste (tlw. in Unterlisten) standen - und zwar in der gleichen Reihenfolge. Wenn keine Liste eingegeben wird, so ist das Ergebnis die leere Liste.

```
?GLAETTE [(1) 2 (3 (3 4) (3 (3 ((4))) 2))]
 Ergebnis = (1 2 3 3 4 3 3 4 2)
?...
```

Wir wollen auch diese Prozedur selber definieren.

```
PR GLAETTELISTE :L
 WENN NICHT LISTE? :L DANN RG []
 SETZE "AKKU []
 GLAETTE1 :L
 RG UMKEHR :AKKU
ENDE
PR GLAETTE1 :L
 WENN :L = [] DANN RK
 TESTE LISTE? :L
 WF SETZE "AKKU ME :L :AKKU RK
 GLAETTE1 ER :L
 GLAETTE1 OE :L
ENDE
```

```
?GLAETTELISTE [(A) (A (B (C) ()) C (D) (((E))))]
 Ergebnis = (A A B C C D E)
?...
```

Diese Prozedur ist recht schwer zu verstehen. Es wird wieder ein AKKU benutzt. Diesmal ist der AKKU aber kein lokaler, sondern ein globaler Name. Dies ist hier notwendig, weil die Prozedur GLATTE1 mit einer Rückgabe nur sehr schwer zu schreiben ist.

Man kann allerdings den Namen AKKU auch lokal definieren. Nur würde dies bedeuten, noch eine Prozedur mehr zu benötigen:

```
PR GLAETTELISTE1 :L
 WENN NICHT LISTE? :L DANN RG []
 RG UMKEHR GLAETTELI :L []              ; Der AKKU wird lokal definiert
ENDE
PR GLAETTELI :L :AKKU
 GLAETTE1 :L
 RG :AKKU
ENDE

PR GLAETTE1                             ; keine Veränderungen!
 WENN :L = [] DANN RK
 TESTE LISTE? :L
 WF SETZE "AKKU ME :L :AKKU RK
 GLAETTE1 ER :L
 GLAETT1 OE :L
ENDE
```

Hier wurde die Prozedur GLAETTELI nur zwischengeschoben, um zu erreichen, daß der Name AKKU nur lokal definiert werden kann.
Versuchen Sie mal, die Prozeduren GLAETTELISTE und GLAETTELISTE1 genau zu betrachten und zu verstehen.

6.4.6 Die Befehle LISTEOHNE und LISTEOHNEALLE

- LISTEOHNE (kurz: LO)
- LISTEOHNEALLE (kurz: LOA)

```
?DZ LISTEOHNE "BAUM [WIESE BAUM KAEFER BAUM SEEROSE]
WIESE KAEFER BAUM SEEROSE
?DZ LISTEOHNEALLE "BAUM [WIESE BAUM KAEFER BAUM SEEROSE]
WIESE KAEFER SEEROSE
?...
```

Der Befehl LISTEOHNE entfernt das als erste Eingabe gegebene Objekt einmal aus der als zweite Eingabe gegebenen Liste (dazu muß das Objekt in der Liste als Element der obersten Ebene erscheinen). Wenn das Objekt nicht in der Liste erscheint, wird die Liste unverändert zurückgegeben. Wenn als zweite Eingabe keine Liste gegeben wird, ist das Ergebnis die leere Liste.
Der Befehl LISTEOHNEALLE entfernt das Objekt (erste Eingabe) aus der Liste (zweite Eingabe) so oft, wie es dort (auf der oberstewn Ebene) als Element auftritt.

Wir wollen auch diese Befehle selber programmieren!

```
PR LISTEOHNE1 :EL :L
 WENN NICHT LISTE? :L DANN RG []
 RG LISTEOHN :L
ENDE
```

```
PR LISTEOHN :L
 WENN :L = [] DANN RUECKGABE :L
 WENN (ER :L) = :EL DANN RG OE :L
 RG ME (ER :L) (LISTEOHN OE :L)
ENDE

PR LISTEOHNEALLE1 :EL :LI
 WENN NICHT LISTE? :L DANN RG []
 RG LISTEOHNEAL :L
ENDE
PR LISTEOHNEAL :L
 WENN :L = [] DANN RUECKGABE :L
 WENN (ER :L) = :EL DANN RG LISTEOHNEAL (OE :L)
 RG ME (ER :L) (LISTEOHNEAL OE :L)
ENDE
```

An den Prozeduren LISTEOHN und LISTEOHNEAL ist noch einmal folgendes gut zu sehen: Der Name :L wird aus der eigenen Privatbibliothek von LISTEOHN genommen, während der Name :EL aus der von LISTEOHNE1 übernommen wird. :EL braucht von LISTEOHNE1 nicht an LISTEOHN übergeben zu werden, weil der Wert dieses Namens immer konstant bleibt.

6.4.7 Die Frage ELEMENT?

- ELEMENT?

```
?ELEMENT? "BAUM [TORTE BAUM SEEROSE]
 Ergebnis = WAHR
?ELEMENT? "HUND [TORTE BAUM SEEROSE]
 Ergebnis = FALSCH
?...
```

Die Frage ELEMENT? prüft ob die erste Eingabe als Element in der Liste (zweite Eingabe) vorhanden ist. Diesen Befehl können wir sehr einfach programmieren:

```
PR ISTELEMENT? :EL :L
 WENN :L = [] DANN RG FALSCH
 WENN (ER :L) = :EL DANN RG WAHR
 RG ISTELEMENT? :EL (OE :L)
ENDE

?DZ ISTELEMENT? 3 [1 2 3 4]
WAHR
?DZ ISTELEMENT? 5 [1 2 3 4]
FALSCH
?...
```

Und nun sind Sie an der Reihe, neue listenverarbeitende Befehle zu erfinden und zu programmieren. Diese Beispiele sind Ihnen hoffentlich eine gute Hilfe gewesen, die Technik des programmierens mit Listen zu verstehen.

7 Interaktive Programme, Ein- und Ausgabe

Unter interaktiven Programmen versteht man, daß ein Programm - einmal gestartet - immer weiter allein arbeiten kann, sich evt. unter neuen Bedingungen wiederholt.

7.1 Schleifen in Prozeduren

Um eine interaktive Prozedur zu schreiben, ist es nötig, daß sich diese Prozedur (oder ein Teil davon) wiederholen kann, ohne daß sie von außen neu aufgerufen werden muß. Wenn sich die Prozedur selbst aufrufen würde, würde sich normalerweise - wir werden eine Ausnahme kennenlernen - die Aufrufebene immer weiter vertiefen, und der Systemstapel (Stack) würde möglicherweise sehr schnell überlaufen. Die einfachste Lösung dieses Problems sind die Merker.

7.1.1 Die Merker

Jede Zeile in einer Prozedur kann mit einem Merker versehen werden. Dieser Merker muß am Anfang der Zeile stehen. Ein Merker ist ein Wort mit einem Doppelpunkt am S c h l u s s (z.B. SCHLEIFE1:). Es kann dann von jeder Zeile dieser Prozedur aus (nicht von anderen Prozeduren aus) in genau diese Zeile "gesprungen" werden. Es ist unerheblich, ob dabei Zeilen übersprungen werden, oder ob zu einer bereits ausgeführten Zeile zurückgesprungen wird. Der "Sprungbefehl" ist

- GEHE.

GEHE braucht ein Wort als Eingabe. Es wird dann in der Prozedur nach einem Merker gesucht, der gleich dem eingegebenen Wort ist.

```
PR WIEDERHOLEDRUCKEN :TEXT
 PUNKT1: DRUCKEZEILE :TEXT
 GEHE "PUNKT1
ENDE

?WIEDERHOLEDRUCKEN "BAUM
BAUM
BAUM
BAUM
BAUM
 ...
```

- Die Ausführung muß mit "Ctrl-C" abgebrochen werden.

Der Merker hat hier den Namen PUNKT1. Bitte verwechseln Sie jetzt nicht Merker mit Namen! Bei Namen muß der Doppelpunkt v o r dem Wort stehen.

Bei der Ausführung der Zeile (PUNKT1: DRUCKZEILE :TEXT) hat der Merker keine Bedeutung. Er wird auch übersprungen, wenn er nicht am Anfang der Zeile steht. Er kann dann aber von AIP - LOGO, bei Aufruf des Befehls GEHE, nicht gefunden werden.

Wenn ein Merker nicht gefunden wird, meldet AIP - LOGO:

```
...
Merker nicht bekannt: PUNKT1:
 in Zeile ...
```

Es wurde der Merker PUNKT1 nicht gefunden.

Der GEHE Befehl verändert nicht die Aufrufebene (Verschachtelungstiefe) und belastet auch nicht den Stapel (auch: Stack = Speicher für Schachtelungen). Es kann eine Prozedur also mit GEHE ewig laufen.

Es ist auch noch anders möglich, eine Prozedur ewig laufen zu lassen:

7.1.2 Selbstaufruf einer Prozedur

```
PR DRUCKEWIG
 DZ [ICH DRUCKE EWIG]
 DRUCKEWIG
ENDE
```

Diese Prozedur DRUCKEWIG ruft sich selbst in der letzten Zeile wieder auf. Dieser "Selbstaufruf" einer Prozedur in ihrer letzten Zeile (es darf auch nichts mehr hinter diesem Aufruf folgen!) ist in AIP - LOGO so effizient implementiert, daß dabei weder die Aufrufebene noch der Stack belastet werden. Dabei ist es auch möglich, bei jedem Selbstaufruf neue Eingaben mitzuübergeben. Trotzdem wird der Stack nicht belastet, und die Aufrufebene bleibt konstant. Die folgende Prozedur ZAEHLEEWIG zählt ununterbrochen, wobei sie sich in ihrer letzten Zeile jeweils selbst mit einem um 1 erhöhten Wert aufruft.

```
PR ZAEHLEEWIG :ZAHL
 DZ :ZAHL
 ZAEHLEEWIG :ZAHL + 1
ENDE
```

7.1.3 Schleifen mit Bedingungen

Erinnern wir uns an den Befehl WIEDERHOLE. Hier wurde so oft eine Schleife gedreht, wie es durch die eingegebene Zahl vorgeschrieben war. Wir können den Abbruch der Schleife aber auch von einer Bedingung abhängig machen. Wir wollen, daß etwas so lange wiederholt wird, bis ein Ereignis eingetreten ist. Dazu stehen die folgenden Befehle zur Verfügung:

- TUEBIS
- TUESOLANGE

Ein Beispiel:

```
?SETZE "A 100
?TUEBIS (REST :A 13) = 0 [SETZE "A :A - 1 DZ :A]
99
98
97
96
95
94
93
92
91                    ; 91 ist durch 13 teilbar, REST 91 13 ist gleich null
?...
```

Der Befehl TUEBIS führt die in der zweiten Eingabe gegebene Befehlsliste immer wieder aus, bis die erste Eingabe (die wird jedesmal neu ermittelt) WAHR geworden ist. Die Ausführung von (SETZE "A :A - 1 DZ :A) wird also so oft wiederholt, bis die Bedingung ((REST :A 13) = 0) WAHR ist.

Ebenso arbeitet der Befehl TUESOLANGE. Es wird die Befehlsliste (zweite Eingabe) so lange ausgeführt, wie die Bedingung (erste Eingabe) erfüllt ist.

7.2 Eingaben über die Tastatur

Es kommt sehr oft vor, daß wir innerhalb einer Prozedur von der Tastatur eine Zeile einlesen oder eine Taste abfragen wollen. Dazu gibt es verschiedene Befehle.

7.2.1 Der Befehl EINGABE

Der Befehl

- EINGABE (kurz: EG)

liest eine Zeile von der Tastatur ein.
Dazu wird - wie gewohnt - ein Fragezeichen am Anfang der nächsten Zeile ausgegeben. Hinter diesem Fragezeichen blinkt der Blinker und AIP - LOGO wartet auf eine Eingabe. Es besteht also kein Unterschied zum Eingeben einer Zeile im Befehls-Modus. Die Zeile wird durch RETURN abgeschlossen. Die Zeile - auf der der Blinker dann steht - wird von AIP -LOGO als Liste (die Liste enthält alle Wörter, Zahlen und Listen, die in der Zeile stehen) zurückgegeben.

Dazu zwei Beispiele:

```
?DZ [NENNE MIR EIN TIER!] EINGABE          ; Befehls-Modus
NENNE MIR EIN TIER!
?MAUS                                      ; EINGABE - Zeile
 Ergebnis = (MAUS)
?DZ OE EINGABE                             ; Befehls-Modus
?BAUM TORTEN HAUS                          ; EINGABE - Zeile
TORTEN HAUS
?...                                       ; Befehls-Modus
```

7.2.2 Tastenabfragen

Der Befehl

- TASTE

veranlaßt AIP - LOGO, auf das Drücken einer Taste zu warten. Dabei blinkt der Blinker. Wenn dann eine Taste gedrückt wird, so wird diese nicht auf dem Bildschirm ausgegeben. Das Zeichen der gedrückten Taste wird als Wort zurückgegeben.

```
?TASTE                          ; der Blinker wartet jetzt auf eine Taste
 Ergebnis = A                   ; Taste "A" wurde gedrückt.
?...
```

Bei diesem Befehl TASTE wird immer auf einen Tastendruck gewartet.
Wenn wir nun wissen wollen, ob im Moment eine Taste gedrückt ist, so können wir dies mit der Frage:

- TASTE?

Ein Beispiel:

```
PR DRUCKET
 WENN TASTE? DANN DRUCKEZEILE "T
 DRUCKET
ENDE

?DRUCKET
T
T
T
 ...
```

Es erscheint kein Blinker auf dem Bildschirm. Trotzdem wird die Tastatur ständig abgefragt. Wenn wir nun eine Taste drücken, werden große T's ausgegeben. Diese Prozedur muß mit "Ctrl-C" abgebrochen werden.
Die Prozedur DRUCKET fragt immer ab, ob eine Taste gedrückt ist, und druckt - wenn ja - ein großes "T" aus. Die Prozedur DRUCKET ruft sich in ihrer letzten Zeile selbst wieder auf und wiederholt sich somit ständig.

Es gibt noch einen dritten Befehl zur Tastaturabfrage:

- TASTEO

Dieser Befehl gibt den Zahlenwert (ASCII-code) einer gedrückten Taste zurück. Wenn im Moment keine Taste gedrückt ist, wird null zurückgegeben. Wir wollen jetzt ein Programm schreiben, in dem der Igel durch das Drücken von Tasten bewegt werden kann. (Der Befehl ASC berechnet den ASCII-Code zu einem Zeichen. Siehe dazu Kap. 7.4.1)

```
PR BEWEGE.IGEL
 BILD STARTIGEL
 BEWEGE.IGEL1
ENDE

PR BEWEGE.IGEL1
 SETZE "NR TASTEO
 WENN :NR = ASC "V DANN VORWAERTS 5
 WENN :NR = ASC "Z DANN RUECKWAERTS 5
 WENN :NR = ASC "R DANN RECHTS 15
 WENN :NR = ASC "L DANN LINKS 15
 BEWEGE.IGEL1
ENDE

?BEWEGE.IGEL
 ...
```

Die Prozedur BEWEGEIGEL startet die Igelgrafik und ruft dann die Prozedur BEWEGE.IGEL1 auf. Die Prozedur BEWEGE.IGEL1 führt für die Buchstaben V, Z, R und L die Befehle VORWAERTS, RUEKWAERTS, RECHTS und LINKS aus. Durch ihre letzte Zeile ruft sich die Prozedur BEWEGE.IGEL< immer wieder selbst auf. Das Programm kann wieder nur durch "Ctrl-C" abgebrochen werden.

Sie sollten jetzt versuchen, diese Prozedur BEWEGE.IGEL und BEWEGE.IGEL1 noch zu verbessern, indem auch auf andere Tasten hin noch Befehle ausgeführt werden. Z.B. könnten die Befehle STIFTHOCH und -AB usw. miteingebunden werden.

7.3 Ausgabe auf dem Bildschirm

Wir haben den Befehl

- DRUCKEZEILE

ja schon viel benutzt. Jedesmal, nachdem eine Zeile gedruckt wurde, wird eine neue Zeile begonnen - d.h. es wird der Blinker in die nächste Zeile gesetzt.

Es wird ja immer an genau der Stelle auf dem Bildschirm etwas ausgedruckt, an der sich der Blinker befindet. Wir können nun vermeiden, daß der Blinker - nachdem etwas ausgedruckt wurde - in die nächste Zeile gesetzt wird. Wir brauchen nur statt des Befehles DRUCKEZEILE den Befehl

- DRUCKE (kurz: DR)

zu benutzen.
Ein Beispiel:

```
PR DRUCKVERSUCH
 DRUCKEZEILE 1
 DRUCKEZEILE 2
 DRUCKEZEILE 3
 DRUCKE 4
 DRUCKE 5
 DRUCKE 6
ENDE

?DRUCKVERSUCH
1
2
3
4 5 6
?...
```

Der Befehl DRUCKE druckt ein Objekt zwar genau wie DRUCKZEILE, springt aber nicht in die nächste Zeile. So können mehrere DRUCKE-Aufrufe in einer Zeile untergebracht werden. Dabei wird nach jedem Drucken eine Leerstelle gesetzt (sonst würden die drei Zahlen 4, 5 und 6 in unserem Beispiel direkt hintereinander stehen).

Wir können auch erreichen, daß keine Leerstelle mitausgegeben wird. Es wird dann alles direkt hintereinander geschrieben. Hierzu benutzen wir den Befehl:

- DRUCK

Ein Beispiel:

```
PR DRUCKVERSUCH1
 DRUCK 7
 DRUCK 8
 DRUCK 9
ENDE

?DRUCKVERSUCH1
789
?...
```

Hinter die Zahlen 7, 8 und 9 werden in diesem Beispiel keine Leerstellen gedruckt.

7.4 Sonstige Befehle für den Bildschirm

- BLINKER

Der Befehl BLINKER verlangt zwei Zahlen als Eingabe. Die erste bestimmt die Spaltennummer und die zweite die Zeilennummer, wo der Blinker hingesetzt werden soll. Es wird von oben links gezählt. Die Spaltennummer ganz links ist null. Die maximale Spaltennummer richtet sich nach der Anzahl der pro Zeile darstellbaren Zeichen (siehe Anhang Teil I.).
Die Zeilennummer der ersten Zeile ist ebenfalls null. Im Bildmodus können nur die untersten fünf Zeilen angesprungen werden.

- LOESCHETEXT (kurz LT)

Dieser Befehl braucht keine Eingaben. Es wird der ganze Bildschirm gelöscht, und der Blinker an den Anfang der ersten Zeile gesetzt.

7.4.1 Der ASCII-Code

Für jedes Zeichen, das der Rechner auf dem Bildschirm abbilden kann, gibt es eine Zahl - einen Code. Diese Zahl entspicht dem allgemein üblichen ASCII-Code. Sie werden im Anhang dieses Handbuches eine Tabelle des ASCII-Codes finden. Die Buchstaben von A bis Z haben z.B. die Nummern 65 bis 90. Es gibt in AIP - LOGO zwei Befehle, die eine Zahl (32 bis 127) in ein Zeichen bzw. ein Zeichen in eine Zahl überführen.

- ASC

Auf Eingabe eines Wortes wird der ASC-Code des ersten Buchstabens ermittelt und zurückgegeben.

```
?ASC "BAUM
 Ergebnis = 66
?...
```

- ZEICHEN

Auf Eingabe einer Zahl (32 bis 127) wird das entsprechende Zeichen als ein Wort (mit einem Zeichen) zurückgegeben.

```
?ZEICHEN 66
 Ergebnis = B
?...
```

7.5 Sonstige Ausgabebefehle

7.5.1 Der Befehl TON

- TON

Der Befehl TON veranlaßt AIP - LOGO, einen kurzen TON auszugeben. Dieser Befehl braucht keine Eingabe.

7.5.2 Ausgabe an einen Drucker

Alles, was auf dem Bildschirm ausgegeben wird, kann auch zusätzlich über ihren Drucker ausgegeben werden (falls einer angeschlossen ist). Dazu müssen wir den Drucker einschalten und den Befehl

- DRUCKEREIN (kurz DE)

eingeben. Von jetzt an wird alles, was auf dem Bildschirm ausgegeben wird - natürlich nicht die Grafik - auch auf Ihrem Drucker ausgegeben. Dieser Befehl wird durch

- DRUCKERAUS (kurz DA)

wieder rückgängig gemacht. Es wird alles nur noch auf dem Bildschirm ausgegeben. Die beiden Befehle werden (nur) auf dem Bildschirm bestätigt:

Drucker läuft ! und
Drucker ist aus !

Wenn kein Drucker angeschlossen, bzw. der Drucker nicht eingeschaltet ist, meldet AIP - LOGO:

Drucker ist nicht bereit !

Diese Meldung führt aber nicht zum Abbruch des Programms! Es wird nur die Druckerausgabe wieder abgeschaltet (wie DRUCKERAUS).

Der Drucker kann auch während der Programmausführung und sogar während einer Auslistung ein- oder ausgeschaltet werden. Drücken Sie dazu:

- "Ctrl-P"

Dieses schaltet auch während der Ausführung eines Programmes immer zwischen DRUCKEREIN und DRUCKERAUS um. So können Sie erreichen, daß nur bestimmte Teile von Auslistungen auf Ihrem Drucker mitgelistet werden.

7.5.3 Die Joystickabfragen

An Ihren Computer können zwei Joysticks angeschlossen werden.

Der Befehl

- STEUER

braucht als Eingabe eine 1 oder 2 (für Joystick 1 oder Joystick 2). Als Ergebnis wird eine Zahl von 0 bis 10 zurückgegeben. Die Bedeutung der einzelnen Zahlen sind in der Tabelle 7.1 aufgeführt.

Die Frage

- KNOPF?

braucht - wie auch STEUER - als Eingabe die Zahlen 1 oder 2 (Joystick 1 oder Joystick 2).

Wenn der Knopf (Feuerknopf) am Joystick gedrückt wird, ist das Ergebnis WAHR, wenn nicht, FALSCH.

(Tabelle 7.1)

Zahlencode	Stellung des Steuers (Joysticks)
0	Mitte (Nullstellung)
1	nach vorne
2	nach hinten
3	*
4	nach links
5	nach links vorne
6	nach links hinten
7	*
8	nach rechts
9	nach rechts vorne
10	nach rechts hinten

* Die Zahlen 3 und 7 werden nicht erzeugt

Im folgenden wird ein ganz kleines Programm gezeigt, wo der Igel mit dem Joystick bewegt werden kann.

```
PR JOYIGEL
 BILD STARTIGEL
 JOYIGEL1
ENDE

PR JOYIGEL1
 SETZE "R STEUER 1
 WENN :R = 1 DANN RI 0
 WENN :R = 2 DANN RI 180
 WENN :R = 4 DANN RI 270
 WENN :R = 5 DANN RI 315
 WENN :R = 6 DANN RI 225
 WENN :R = 8 DANN RI 90
 WENN :R = 9 DANN RI 45
 WENN :R = 10 DANN RI 135
 WENN NICHT :R = 0 DANN VW 10
 WENN KNOPF? 1 DANN RUECKKEHR
 JOYIGEL1
ENDE
```

Der Igel kann jetzt mit dem Joystick (der Joystick muß in der Anschlußbuchse für Joystick 1 stecken!) bewegt werden.

7.6 Die F-Tasten

Auch in AIP - LOGO ist es möglich, den F-Tasten (Funktions-Tasten) ihres Computers (siehe Kapitel 1.3.2) Wörter oder Listen zuzuweisen. Dies geschieht mit dem Befehl

- FTASTE.

Dieser Befehl braucht zwei Eingaben. Die erste bezeichnet die Nummer der F-Taste (1 - 10) und die zweite enthält das Wort oder die Liste, die der zugewiesen werden soll.
Abfragen, welcher F-Taste was zugewiesen wurde, können Sie mit dem Befehl

- FTASTEN.

Dazu Beispiele:

```
?FTASTE 1 "BILD
?FTASTE 2 "SETZE
?FTASTE 3 [LOGO IST DOOF]
?...
```

Immer, wenn Sie jetzt in einer Eingabezeile die F-Taste 1 drücken, wird der Befehl "BILD" in der Zeile ausgegeben. Bei F-Taste 2 wird der Befehl "SETZE" und bei F-Taste 3 der Satz "LOGO IST DOOF" ausgegeben.

Der Befehl FTASTEN listet alle Zuweisungen der F-Tasten aus:

```
?FTASTEN
 1 BILD
 2 SETZE
 3 LOGO IST DOOF
 4
 5
 6
 7
 8
 9
10
?...
```

Sie können sich so die Wörter und Befehle auf die F-Tasten legen, die Sie besonders häufig benutzen.

8 Programmkontrolle, Fehlerbehandlung

AIP - LOGO stellt eine Menge Möglichkeiten zur Verfügung, den Programmablauf zu kontrollieren und Fehler zu finden. Als erstes soll die Verwendung von runden und Eckigen Klammern in AIP - LOGO noch einmal ganz genau beschrieben werden.

8.1 Der Gebrauch von Klammern in AIP - LOGO

Wir können Teile von Befehlszeilen in Klammern zusammenfassen:

- Bei mathematischen Ausdrücken:

```
z.B.    DZ (3 + 5) * 2
und     DZ 3 + 5 * 2
```

Das Ergebnis des ersten Ausdrucks ist 30, das des zweiten 13. So können wir durch das Zusammenfassen mathematischer Begriffe erreichen, daß erst der Inhalt der Klammern ausgerechnet wird, bevor nach der "Punkt vor Strich" Regel weitergerechnet wird.

- Zur übersichtlichkeit (ohne Wirkung):

Bei langen Befehlszeilen ist es häufig sinnvoll, Teile der Zeile in Klammern zusammenzufassen, damit klar wird, welche Eingabe(n) zu welchem Befehl gehören.

```
?DZ ME OE "BAUM ME LISTE "PETRA "TORTEN []
NJA (PETRA TORTEN)
?DZ (ME (OE "BAUM) (ME (LISTE "PETRA "TORTEN) [])
NJA (PETRA TORTEN)
?...
```

In der zweiten Zeile sind nun einige Teile in Klammern gefaßt. Dies ändert die Ausführung der Zeile allerdings nicht. Wir dürfen selbstverständlich nur solche Teile zusammenfassen, die auch logisch zusammengehören. Wenn wir die Zeile anders klammern würden, könnten Fehler entstehen.

```
?DZ ME (OE "BAUM ME) LISTE "PETRA "TORTEN []
 zuviel in (...) : (OE "BAUM ME)
?...
```

Hier wurden willkürlich zwei Klammern in die Zeile eingefügt. AIP - LOGO denkt nun, daß sowohl "BAUM als auch ME Eingaben für OE sind. Da AIP - LOGO aber keine zwei Eingaben für OHNEERSTES gebrauchen kann, wird der Fehler gemeldet.

Das Klammern von Begriffen hat in AIP - LOGO noch eine andere Bedeutung. Auch diese haben wir schon im Zusammenhang mit vielen Befehlen kennengelernt. Normalerweise braucht der Befehl LISTE zwei Eingaben, die als zwei Elemente in einer neuen Liste untergebracht werden.

```
?DZ LISTE "BAUM "HAUS
BAUM HAUS
?...
```

Wenn wir diesen Befehl nun in Klammern schreiben, so ist die Zahl der Eingaben beliebig. Es wird alles, was noch in der Liste (in den Klammern) steht, mit als Eingabe für den Befehl LISTE genommen und in der neuen Liste untergebracht.

```
?DZ (LISTE "BAUM)
BAUM
?DZ (LISTE "BAUM "HAUS "TORTEN "MAUS)
BAUM HAUS TORTEN MAUS
?DZ (LISTE)
                                    ;leere Zeile
?...
```

Es kann der Befehl LISTE jetzt sogar ohne Eingaben aufgerufen werden. Das gilt für viele Befehl in AIP - LOGO. Diese besondere Eigenschaft der Befehle wurde und wird jedesmal - wenn ein Befehl neu eingeführt wird - mitaufgezeigt.

8.2 Das Fehlermeldesystem von AIP - LOGO

Das Fehlermeldesystem von AIP - LOGO besteht aus zwei Teilen. Zum einen aus einer detallierten Mitteilung über Art und Ort des aufgetretenen Fehlers, zum anderen aus dem Fehlermodus (Errormodus), der aber erst in Kapitel 8.3 untersucht werden soll.

Sowie AIP - LOGO auf einen Fehler stößt, wird vom Fehlermeldesystem die Art des aufgetretenen Fehlers (außerdem evt. der Befehl, bei dessen Aufruf der Fehler aufgetreten ist), sowie die Zeile und die Prozedur, in der der Fehler aufgetreten ist, ausgegeben.
Hierzu ein Beispiel: Wir definieren uns folgende - zugegeben, nicht sehr sinnvolle - Prozedur FEHLERTEST und rufen diese mit der Eingabe 3 auf:

```
PR FEHLERTEST :A
 WENN :A = 0 DANN LALALA
 FEHLERTEST :A - 1
 DZ "FERTIG
ENDE
```

```
?FEHLERTEST 3
Prozedur LALALA nicht bekannt !                    ; Mitteilung 1
  in Zeile : WENN :A = 0 DANN LALALA               ; Mitteilung 2
 in Aufrufebene 4 von: FEHLERTEST                  ; Mitteilung 3
E4?...
```

Anmerkung: In diesem Beispiel tritt der Fehler erst auf, wenn die Bedingung A = 0 erfüllt (WAHR) ist und somit die Prozedur LALALA ausgeführt werden soll. Bis nun der Name :A den Wert null hat, muß sich die Prozedur FEHLERTEST dreimal selbst aufgerufen haben. Damit befindet sich das Programm in der Aufrufebene 4 (Schachtelungstiefe 4).

Die drei Mitteilungen des AIP - LOGO Fehlermeldesystems werden jetzt genauer untersucht.

Die Mitteilung 1 enthält die Art des aufgetretenen Fehlers. Diese Mitteilung wird bei jedem Fehler gegeben. Sie kann teilweise auch noch den Namen einer Prozedur oder eines Befehls enthalten, bei dessen Aufruf der Fehler aufgetreten ist. Z.B.

```
?WH 2.3 [DZ 1]
 Ganze Zahl erwartet von : WH
?...
```

Hier wurde der Befehl WIEDERHOLE mit einer Realzahl aufgerufen. Die Fehlermeldung nennt hier auch den Befehl, der hier eine ganze Zahl als Eingabe gebraucht hätte.
Die meisten Fehlermeldungen finden Sie im Anhang aufgelistet.

Wenn nun ein Fehler aufgetreten ist, wird das Programm grundsätzlich abgebrochen.

Es gibt nur eine Ausnahme:
Wenn der Drucker während des Programms (mit DRUCKEREIN oder "Ctrl-P") angesprochen werden soll, aber nicht angeschlossen oder eingeschaltet ist, wird die Fehlermeldung

```
 Drucker ist nicht bereit!
```

ausgegeben und die Druckeransteuerung wieder abgeschaltet. Das Programm wird aber dann weiter ausgeführt. In allen anderen Fällen wird das Programm abgebrochen.

Die Mitteilung 2 zeigt die Zeile, in der der Fehler aufgetreten ist. Dabei wird auch angegeben, wenn es sich um eine besondere Art von Zeile (z.B. eine WIEDERHOLE-Zeile) handelt.

Ein Beispiel:

```
?WIEDERHOLE 4 [BLABLABLA 123]
Prozedur BLABLABLA nicht bekannt !
  in WIEDERHOLE - Zeile : BLABLABLA 123
E0?...
```

Hier ist der Fehler innerhalb der Befehlsliste eines WIEDERHOLE - Aufrufs aufgetreten. Es wird deutlich, daß AIP - LOGO diese Befehlsliste als eigenständige Befehlszeile ansieht.
Außer dem Befehl WIEDERHOLE bauen sich die Befehle TUESOLANGE, TUEBIS, und TUE eigene Befehlszeilen auf. Wenn innerhalb dieser Befehlszeilen ein Fehler auftritt, wird dieses hier besonders vermerkt.
Die Mitteilung 2 wird nur mitausgegeben, wenn der Fehler nicht in der eingegebenen Befehlszeile auftgetreten ist. Sie können davon ausgehen, daß, wenn in der Fehlermeldung nicht angegeben wird, in welcher Zeile der Fehler aufgetreten ist, diese Zeile die eben eingegebene Befehlszeile ist.

Die Mitteilung 3 beschreibt die Aufrufebene sowie die Prozedur, in der der Fehler aufgetreten ist. Diese wird natürlich nur ausgegeben, wenn der Fehler innerhalb einer Prozedur liegt ist. Diese dritte Mitteilung ist immer von gleichen Art:

```
in Aufrufebene nn von NAMENAME                    ; Mitteilung 3
```

nn ist die Aufrufebene (Schachtelungstiefe) und
NAMENAME ist die Prozedur, in der der Fehler aufgetreten ist.

Wenn keine Mitteilung 2 und 3 gemacht wurde, wird direkt in den Befehlsmodus zurückgesprungen. Wenn diese beiden Mitteilungen gemacht wurden, wird in den Fehlermodus (E=Errormodus) übergegangen.

8.3 Der Fehler- und der Unterbrechungsmodus

Wenn innerhalb einer Prozedur oder einer WIEDERHOLE - Zeile etc. ein Fehler aufgetreten ist, wird - nachdem eine detaillierte Fehlermeldung ausgegeben wurde - in den Fehlermodus übergegangen.

Nach Betätigung von

- "Ctrl-Z"

während der Programmausführung geht AIP - LOGO in den Unterbrechungsmodus über. Zuvor wird noch der genaue Ort der Unterbrechung angegeben. Diese Angaben entsprechen vollständig den Mitteilungen 2 und 3 der Fehlermeldungen (s. Kap. 8.2).

Sie können die beiden Modi durch die Buchstaben vor dem Fragezeichen in jeder Eingabezeile erkennen.

```
I3?...                    ; Unterbrechungsmodus (Aufrufebene 3)

E0?...                    ; Fehler- oder Errormodus (Aufrufebene 0)
```

Die Aufrufebene 0 ist die Eingabeebene. Wenn aber ein Fehler in der eingegebenen Befehlszeile auftritt, oder die Programmausführung dort mit "Ctrl-Z" abgebrochen wird, geht AIP - LOGO sofort wieder in den Befehlsmodus über. Nur wenn der Fehler oder die Unterbrechung in z.B. einer WIEDERHOLE-Zeile innerhalb der eingegebenen Befehlszeile auftritt, wird in den Fehler- oder Unterbrechungsmodus gesprungen.

Der Unterbrechungsmodus ist fast völlig gleich zum Fehlermodus. Es ist jedoch im Unterbrechungsmodus möglich, das Programm fortzusetzen. Dazu wird der Befehl

- WEITER (kurz: WT)

verwendet. Dann wird das Programm weiterausgeführt.

Wir wollen jetzt etwas mit dem Fehlermodus arbeiten. Geben wir ein:

```
PR FEHLERVERSUCH :A
 WENN :A = 0 DANN BLABLA
 FEHLERVERSUCH :A - 1
 DZ "FEHLERVERSUCH
ENDE

?FEHLERVERSUCH 5
Prozedur BLABLA nicht bekannt !
  in Zeile : WENN :A = 0 DANN BLABLA
 in Aufrufebene 6 von: FEHLERVERSUCH
E6?...
```

Nun können wir etwas ausprobieren. Geben wir jetzt ein:

```
E6?DZ :A
0
E6?...
```

Im Befehlsmodus wäre Name :A als lokaler Name innerhalb der Prozedur FEHLERVERSUCH nicht bekannt. Im Fehlermodus jedoch, ist er jetzt bekannt. Das erklärt sich dadurch, daß sich AIP - LOGO noch innerhalb der Prozedur FEHLERVERSUCH befindet. Alle Namen sind wie in dieser Prozedur bekannt. Dies ermöglicht bei komplizierten Fehlern das Erforschen der Umgebung, in der der Fehler aufgetreten ist.

Sie erinnern sich noch: Durch "Ctrl-C" oder durch den Befehl AUSSTIEG (kurz: AS) kann vom Fehler- oder Unterbrechungsmodus zurück zum Befehlsmodus gelangt werden.

Im Unterbrechungsmodus können wir auch Namen verändern und dann das Programm weiterausführen lassen. Wir können also in das Programm eingreifen und somit auch den Ablauf damit beeinflussen. Dazu ein einfaches Beispiel:

```
PR ZAEHLE :ANFANG
 DZ :ANFANG
 ZAEHLE :ANFANG + 1
ENDE

?ZAEHLE 0
0
1
2
3                                ; "Ctrl-Z"
 ** Unterbrechung !
  in Zeile DZ :ANFANG
 in Aufrufebene 1 von: ZAEHLE
I1?DZ :ANFANG
3
I1?SETZE "ANFANG 100
I1?WEITER
101
102
103
 ...
```

Wir haben das Programm durch "Ctrl-Z" angehalten, den Wert des lokalen Namens :ANFANG geändert und das Programm dann weiter ausgeführt. So können größere Programme auch gut getestet werden.

Wenn wir nun an einer ganz bestimmten Stelle der Prozedur in den Unterbrechungsmodus gelangen wollen, um z.B. die Werte von einigen Namen zu überprüfen, so können wir dies mit folgendem Befehl erreichen:

- HALT

Wenn AIP - LOGO innerhalb einer Prozedurzeile auf diesen Befehl trifft, wird sofort in den Unterbrechungsmodus gesprungen. Selbstverständlich kann auch hier dann die Prozedur durch den Befehl WEITER fortgesetzt werden.

- PAUSE

Dieser Befehl veranlaßt AIP - LOGO, auf einen Tastendruck zu warten. Dabei blinkt der Blinker. Wenn eine Taste gedrückt wird, wird das Programm weiter ausgeführt. Das Zeichen der Taste wird nicht ausgedruckt. An dieser Stelle kann natürlich das Programm mit "Ctrl-C" abgebrochen und mit "Ctrl-Z" unterbrochen werden. Ebenso können auch die anderen "Ctrl-..." Tasten benutzt werden.

- AUSSTIEG (kurz: AS)

Dieser Befehl veranlaßt AIP - LOGO, die Programmausführung vollständig abzubrechen. AIP - LOGO kehrt in den Befehlsmodus zurück. Dieser Befehl ist vergleichbar mit "Ctrl-C" während der Programmausführung.

Ein Programm kann mit einer Zeitschleife verzögert werden:

- WARTE

```
?WARTE 100
?...
```

Der Befehl WARTE braucht eine ganze Zahl (0 bis 32767) als Eingabe. AIP - LOGO wartet dann eine gewisse Zeit, bevor das Programm weiterausgeführt wird. Die Eingabe gibt die Zeit an (in 1/100 Sekunden). In dem Beispiel oben wird eine Sekunde gewartet.

8.3.1 Der Fehlereditor EDF

Wenn AIP - LOGO innerhalb einer Prozedur auf einen Fehler stößt, wird dieser nicht nur genau beschrieben (usw.), sondern es wird auch dem Editor von AIP - LOGO die Prozedur und Zeile - wo der Fehler auftrat - mitgeteilt. Durch den Befehl

- EDF

ist es möglich, in den Editor zu springen, ohne den Namen der fehlerhaften Prozedur einzugeben. Es wird die Prozedur angenommen, in der zuletzt ein Fehler gemeldet wurde. Ebenso wird auch gleich in die fehlerhafte Zeile gesprungen. Dazu ein kleines Beispiel:

```
PR FTEST
 DZ 1
 FEHLERFEHLER
 DZ 2
ENDE

?FTEST
1
Prozedur FEHLERFEHLER nicht bekannt !
  in Zeile : FEHLERFEHLER
 in Aufrufebene 1 von: FTEST
E1?EDF
      PR FTEST
1:    DZ 1
2:    FEHLERFEHLER
3:    DZ 2

2:    FEHLERFEHLER
 ...
```

Nachdem der Fehler in der zweiten Zeile der Prozedur FTEST gemeldet wurde, springt AIP - LOGO auf den Befehl "EDF" in den Editor an genau diese Stelle. Diese Möglichkeit erleichtert das Korrigieren von Fehlern in Prozeduren erheblich.

8.4 Das Ablaufprotokoll in AIP - LOGO

In AIP - LOGO ist ein Ablaufprotokoll verwirklicht, das mit folgenden drei Befehlen gesteuert werden kann:

- PROTOKOLLEIN (kurz: PE)
- PROTOKOLLAUS (kurz: PA)
- BETRIEB (kurz: BTR)

Eingeschaltet wird das Ablaufprotokoll durch den Befehl PROTOKOLLEIN, wieder ausgeschaltet wird es durch den Befehl PROTOKOLLAUS. Ebenso kann das Ablaufprotokoll aber auch - wie Sie es auch von der parallelen Druckerausgabe gewohnt sind - durch "Ctrl-Q" ein- und ausgeschaltet werden. Dies ist selbstverständlich auch während der Programmausführung möglich.

Wird das Ablaufprotokoll per Befehl (nicht durch "Ctrl-Q") ein- oder ausgeschaltet, wird dies durch eine Meldung des Systems bestätigt:

```
Protokoll laeuft !
oder
Protokoll ist aus !
```

Nun zum Aufbau des Ablaufprotokolls.

```
?PROTOKOLLEIN
 Protokoll laeuft !
?DZ "VERSUCH
VERSUCH
?WIEDERHOLE 4 [DZ "VERSUCH]
 WIEDERHOLE - Zeile : DZ "VERSUCH
VERSUCH
 WIEDERHOLE - Zeile : DZ "VERSUCH
VERSUCH
 WIEDERHOLE - Zeile : DZ "VERSUCH
VERSUCH
 WIEDERHOLE - Zeile : DZ "VERSUCH
VERSUCH
?...
```

Direkt eingegebene Zeilen werden nicht protokolliert. WIDERHOLE - Zeilen, TUE - Zeilen usw. werden auch innerhalb direkt eingegebener Befehlszeilen protokolliert.
Es werden also alle Zeilen (nicht die direkt eingegebenen Zeilen) ausgedruckt, bevor sie ausgeführt werden. Dabei wird mitangegeben, um was für eine Zeile (einfache Zeile oder WIEDERHOLE - Zeile usw.) es sich handelt:

```
Zeile : ...
WIEDERHOLE - Zeile : ...
TUE - Zeile : ...
TUESOLANGE - Zeile : ...
TUEBIS - Zeile : ...
```

Die Entfernung dieser Meldung vom linken Bildschirmrand richtet sich nach der Aufrufebene. In Aufrufebene 1 sind drei Leerstellen dazwischen, in Aufrufebene 5 sind es 7 Leerstellen usw.. Wenn eine Prozedur aufgerufen wird, wird dieses vom Ablaufprotokoll folgendermaßen beschrieben:

```
PR DRUCKEEINEEINS
 DZ 1
ENDE

?PROTOKOLLEIN
 Protokoll laeuft !
?DRUCKEEINEEINS
   E1: Aufruf der Prozedur : DRUCKEEINEEINS
   Zeile : DZ 1
1
   E1: Ende : DRUCKEEINEEINS
?...
```

Wenn beim Aufruf der Prozedur lokale Namen besetzt werden, werden diese mit den ihnen zugewiesenen Werten (Objekten) alle ausgelistet. Ebenso wird beim Ende der Prozedur ein eventuelles Ergebnis ausgegeben:

```
PR APLUS2B :A :B
 RUECKGABE :A + 2 * :B
ENDE

?PE
 Protokoll laeuft !
?DZ APLUS2B 3 5
   E1: Aufruf der Prozedur : APLUS2B
    :A = 3
    :B = 5
   Zeile : RUECKGABE :A + 2 * :B
   E1: Ende : APLUS2B = 13
13
?...
```

Es werden die besetzten lokalen Namen der Prozedur in der Reihenfolge ihrer Besetzung ausgelistet. Ebenso das Ergebnis der Prozedur (13).

Wenn sich eine Prozedur immer mehr verschachtelt, werden die Meldungen immer weiter eingerückt. Dies gilt selbsverständlich nicht, wenn sich die Prozedur in ihrer letzten Zeile selbst aufruft, da die Aufrufebene dann nicht verändert wird. Dazu zwei Beispiele von zählenden Prozeduren:

```
PR ZAEHLE1 :A
 WENN :A = 0 DANN RUECKKEHR
 DZ :A
 ZAEHLE1 :A - 1
ENDE
```

```
?PE
 Protokoll laeuft !
?ZAEHLE1 3
   E1: Aufruf der Prozedur : ZAEHLE1
    :A = 3
   Zeile : WENN :A = 0 DANN RUECKKEHR
   Zeile : DZ :A
3
   Zeile : ZAEHLE1 :A - 1
   E1: Aufruf der Prozedur : ZAEHLE1
    :A = 2
   Zeile : WENN :A = 0 DANN RUECKKEHR
   Zeile : DZ :A
2
   Zeile : ZAEHLE1 :A - 1
   E1: Aufruf der Prozedur : ZAEHLE1
    :A = 1
   Zeile : WENN :A = 0 DANN RUECKKEHR
   Zeile : DZ :A
1
   Zeile : ZAEHLE1 :A - 1
   E1: Aufruf der Prozedur : ZAEHLE1
    :A = 0
   Zeile : WENN :A = 0 DANN RUECKKEHR
   E1: Ende : ZAEHLE1
?...
```

Hier verändert sich die Aufrufebene nicht! Das Ablaufprotokoll zeigt, daß die Prozedur ZAEHLE1 immer in der Aufrufebene 1 ausgeführt wird. Wenn wir nun aber eine Prozedur schreiben, die sich richtig immer tiefer verschachtelt, so rücken auch die Meldungen des Protokolls immer weiter ein. Dazu folgende Prozedur ZAEHLE2. Diese Prozedur vertieft sich (am Protokoll sehr gut zu verfolgen) erst bis zum Ende und gibt die Zahlen dann in aufsteigender Reihenfolge beim "Wiederhochkommen" aus (rückwärts-rekursiv).

Die Prozedur ZAEHLE2:

```
PR ZAEHLE2 :A
 WENN :A = 0 DANN RUECKKEHR
 ZAEHLE :A - 1
 DZ :A
ENDE
```

Auf der folgenden Seite wird nun ein Ablaufprotokoll dieser Prozedur gelistet:

```
?PE
 Protokoll laeuft !
?ZAEHLE2 3
   E1: Aufruf der Prozedur : ZAEHLE2
    :A = 3
   Zeile : WENN :A = 0 DANN RUECKKEHR
   Zeile : ZAEHLE2 :A - 1
    E2: Aufruf der Prozedur : ZAEHLE2
     :A = 2
    Zeile : WENN :A = 0 DANN RUECKKEHR
    Zeile : ZAEHLE2 :A - 1
     E3: Aufruf der Prozedur ZAEHLE2
      :A = 1
     Zeile : WENN :A = 0 DANN RUECKKEHR
     Zeile : ZAEHLE2 :A - 1
      E4: Aufruf der Prozedur : ZAEHLE2
       :A = 0
      Zeile : WENN :A = 0 DANN RUECKKEHR
      E4: Ende : ZAEHLE2
     Zeile : DZ :A
1
     E3: Ende : ZAEHLE2
    Zeile : DZ :A
2
    E2: Ende : ZAEHLE2
   Zeile : DZ :A
3
   E1: Ende : ZAEHLE2
?...
```

Nun zu dem Befehl BETRIEB. Durch diesen Befehl kann die Ausführung von Zeilen bei eingeschaltetem Ablaufprotokoll verzögert werden. Der Befehl braucht eine Eingabe. Diese Eingabe muß eine Zahl von 0 bis 3 (incl.) sein. Die Zahlen haben folgende Bedeutung:

0 Normale (maximale) Ausführungsgeschwindigkeit.
1 Etwas verzögerte Ausführung bei eingeschaltetem Protokoll.
2 Stark verzögerte Ausführung bei eingeschaltetem Protokoll.
3 Vor Ausführung einer jeden Zeile - nach Ausgabe der Protokollmeldung - wird auf einen Tastendruck gewartet. Dabei blinkt der Blinker, um das Warten anzuzeigen. Nach einem Tastendruck wird die Zeile ausgeführt.

Ein Beispiel:

```
?PROTOKOLLEIN
 Protokoll laeuft !
?BETRIEB 2
?...
```

Nun wird die Ausführung von Zeilen, die auch protokolliert werden, stark verzögert! Probieren Sie die vier Betriebsgeschwindigkeiten einfach einmal aus!

9 Weitergehende Anwendungen von Listen

oder: Erste Schritte zur künstlichen Intelligenz

9.1 Was ist künstliche Intelligenz (KI)

Das erste große Problem der KI ist, daß man sich nicht so recht einig darüber ist, was Intelligenz eigentlich sein soll. Auch der Autor selbst ist bisher nicht zu einer endgültigen Ansicht über die Definition von Intelligenz gekommen. Es wird daher im folgenden versucht, den Begriff der Intelligenz zu umgehen.

- Eine viel gewünschte (und auch gefürchtete) Form der KI besteht darin, daß ein Programm in der Lage ist, nicht eingeplante Probleme zu lösen. Dazu muß das Programm allerdings zu einer Art von Improvisation fähig sein. Es müßte selbst auf völlig neue (nicht einprogrammierte) Ideen kommen. Es ist bisher (glücklicherweise) nicht endgültig gelungen, ein solches Programm zu erstellen.

- Ebenso stellt sich die interessante Frage, ob es möglich ist, mit einem Computer einen Menschen zu simulieren. Stellen wir uns folgenden Versuch vor:
Es gibt drei von einander schalldicht und telepathiedicht getrennte Räume. Im ersten der drei Räume befindet sich ein Computer, im zweiten ein Mensch. Im dritten Raum befindet sich die Testperson, die nun herausfinden soll, in welchem der ersten beiden Räume der Computer ist und in welchem der Mensch. Der Testperson im dritten Raum stehen nur eine Tastatur und ein Bildschirm zur Verfügung, mit denen sie mit dem Menschen im zweiten und mit dem Computer in dem ersten Raum "reden" kann. Die Person kann nun Fragen an den ersten und den zweiten Raum stellen. Der Mensch im zweiten Raum wird immer versuchen, der Testperson zu zeigen, daß er der Mensch ist. Aber auch der Computer ist so progrmmiert, daß er versucht, die Testperson davon zu überzeugen, daß "er" der Mensch ist. Dieser Versuch wird mit vielen Testpersonen durchgeführt. Der Versuch gilt als erfolgreich, wenn weniger als zwei Drittel der Testpersonen den Menschen vom Coputer richtig unterscheiden.
Der Versuch entspricht der Idee des "Turing-Tests", der bereits 1950 von Alan Turing in einem Artikel beschrieben wurde.
Es ist wohl eine Überforderung Ihres kleinen Coputers, ihn für so einen Test zu programmieren. Dies ist nur auf großen, schnellen Rechenanlagen vernünftig.

- Viele Versuche, die auf dem Gebiet der KI gestartet wurden, basierten auf dem Prinzip der Assoziation. Das bedeutet, mit jedem Ding bzw. der Kombination von Dingen wird ein anderes Ding assoziiert. Wenn das Programm z.B. an TIER und an GROSS "denkt", so sollte es als erstes auf ELEFANT "kommen". Natürlich darf diese Beziehung zwischen großem Tier und Elefant nicht direkt eingegeben werden. Das Programm muß vielmehr in der Lage sein, diese Beziehung selbst zu erkennen und zu lernen.

- Ein weiterer Aspekt der KI ist es, ein Programm so zu gestalten, daß es ein anderes Programm - oder auch sich selbst - verändern kann. Dadurch könnte sich das Programm einer gestellten Aufgabe evt. anpassen.
Stellen wir uns vor, wir wollen ein Programm schaffen, welches spielen kann. Es soll ein Spiel jeweils erst lernen und dann spielen können, wobei es durch das Lernen beim Spiel immer besser werden soll. Das Programm müßte sich dann immer weiter verfeinern. Hier stoßen wir wieder auf die Grenzen, da jedes System nur begrenzt Speicherplatz hat.

- Es wird auch versucht - besonders von Psychologen -, das menschliche "Denken" zu analysieren und nachzuahmen. Dies ist sehr schwierig zu realisieren, da die Vorgänge im menschlichen Gehirn sehr komplex ablaufen, und auch viele Vorgänge dort parallel geschehen, die wir aber im Computer nacheinander ablaufen lassen müssen. Dabei werden die Möglichkeiten des Computers durch die relativ geringen Rechengeschwindigkeiten stark eingeschränkt.

So sind die meisten Programme auf dem Gebiet der KI bisher nur Versuche, mit denen keiner völlig zufrieden ist. Sie erscheinen vielleicht nach außen hin als sehr vielseitig oder gar schöpferisch, arbeiten aber trotzdem noch nach starren (unintelligenten) Ablaufregeln, die der Programmierer vorher eingegeben hat. Hier sind z.B. die Expertensysteme zu nennen, die tlw. in der Lage sind, Bücher - sogenannte "Dreigroschenromane" - selbst zu schreiben. Auch gibt es welche, die Kreuzworträtzel "erfinden" oder die mathematische Gleichungen auflösen können. Selbst für wirtschaftliche oder gar politische Fragen können Expertensysteme programmiert werden, die aus der Verarbeitung von ungeheuer großen Datenmengen eine Entscheidung treffen oder Entscheidungshilfen geben können. Auch wenn all diese Programme nach außen "genial" wirken, arbeiten sie (fast) immer nach einem strengen Schema. Trotzdem bezeichnet man sie manchmal gern als intelligent.

9.2 Anwendung der KI in AIP - LOGO

Sie werden schon an dieser Stelle merken, daß es nicht so ohne weiteres möglich ist, ein "intelligentes" Programm zu schreiben. Vielmehr können zunächst nur Versuche gemacht werden, Bausteine eines KI-Programms zu erstellen.
Dadurch, daß AIP - LOGO listenorientiert ist, ist es relativ einfach möglich, Assoziationen in Form von listenartigen Zuordnungen darzustellen. Es könnte z.B. eine Liste geben, die zu ihrem ersten Element eine Reihe von Zuordnungen oder Assoziationen enthält. Vorteil ist hier in AIP - LOGO, daß das gleiche Objekt (Wort, Zahl oder Liste) mehrmals in Listen auftauchen kann, und dabei nur einmal im Speicher stehen muß. Damit wird viel Speicherplatz gespart.
Auch sind Prozeduren in LOGO in der Lage, andere Prozeduren (auch sich selbst) zu verändern. Das wird erst dadurch möglich, daß die Prozeduren nicht nur im gleichen Bereich des Speichers wie die Daten (Variablen, Namen) stehen, sondern daß sie den Daten auch von der Struktur her gleich sind. Jede Prozedur läßt sich als eine Liste darstellen, und jede Liste als Prozedur definieren oder als TUE-Zeile ausführen.

Die Befehle dazu sollen jetzt in 9.3 und 9.4 erklärt werden.

9.3 Der TUE - Befehl

Der Befehl

- TUE

interpretiert eine Liste als Befehlszeile und führt diese aus.

```
?TUE [DZ "BAUM]
BAUM
?TUE ME "DZ [123 * 2]
246
?...
```

Die TUE-Zeile kann beliebig lang sein. Sie wird wie jede andere Zeile behandelt. Sie kann auch ein Ergebnis haben. Dieses Ergebnis wird dann von TUE weiter zurückgegeben.

```
?3 + TUE [WURZEL 4]
 Ergebnis = 5
?...
```

Wenn in einer Tuezeile ein Fehler auftritt, so wird darauf von AIP - LOGO besonders hingewiesen:

```
?TUE [DRUCKZEILE "BAUM]
Prozedur BAUM nicht bekannt !
  in TUE-Zeile : DRUCKZEILE BAUM
?...
```

Wir werden noch einige Bespiele mit dem Befehl TUE kennenlernen.

9.4 Umwandlungen von Prozeduren und Listen

Wir haben die folgende Prozedur QUADRAT.

```
?ZEIGE "QUADRAT
PR QUADRAT :GROESSE
 WH 4 [VW :GROESSE RE 90]
ENDE
?PRLISTE "QUADRAT
 Ergebnis = ((:GROESSE) (WH 4 [VW :GROESSE RE 90]))
?DZ ERSTES PRLISTE "QUADRAT
:GROESSE                                    ; gemeint ist (:GROESSE)
?...
```

Wir können nun den Prozedurkörper mit dem Befehl PRLISTE (Prozedurnamen als Eingabe) bekommen und weiterverarbeiten. Hierbei wird eine Liste zurückgegeben. Das erste Element dieser Liste ist die Liste der Eingabennamen. Wenn keine Eingaben gefordert werden, so ist dies die leere Liste. Alle weiteren Elemente sind die Zeilen dieser Prozedur. Alle Zeilen sind wieder Listen. Der Inhalt der Zeilen steht nun in diesen Listen. Sollten auch in den Zeilen noch Klammern (Listen) auftauchen, so werden diese natürlich mitaufgeführt.

Ebenso können wir auch eine Liste als Prozedur definieren. Wir brauchen als Eingaben den Prozedurnamen und den Körper in eben genannter Form:

```
?DEF "NEUESQUADRAT [(:GROESSE) (WH 4 [VW :GROESSE RE 90])]
?ZEIGE "NEUESQUADRAT
PR NEUESQUADRAT :GROESSE
 WH 4 [VW :GROESSE RE 90]
ENDE
?...
```

Hiebei können leider sehr viele Fehler auftreten. AIP - LOGO meldet häufig "unsinnige Titelzeile ! ...". Dies bedeutet, wir haben entweder einen nicht erlaubten Namen eingegeben, oder das erste Element - die Liste mit den Eingabenforderungen - ist fehlerhaft (eventuell fehlt sie auch völlig, und AIP - LOGO nimmt die erste Zeile als Liste der Eingaben).
Sollten Sie als Zeilen der Prozedur Unsinn eingeben, so wird dies von LOGO nicht bemerkt, zumindest nicht gleich. Bei der Ausführung der fehlerhaften Prozedur wird AIP - LOGO dann auf den Fehler stoßen und ihn melden. Häufig geschieht dies durch folgende Meldung:

```
Unsinniger Begriff : ()
```

Dann ist meistens eine unsinnige Zeile eingegeben worden.

9.5 Eigenschaftslisten

In AIP - LOGO ist eine vollständige Verwaltung für Eigenschaftslisten verwirklicht.

Eigenschaftslisten sind Listen, die jeweils einem Objekt (Namen) zugeordnet werden. In diesen Listen stehen die Eigenschaften des Objekts. Und zwar in der Form, daß dort zu jeweils einer "Eigenschaftsbezeichnung" eine "Beschreibung" steht.

Für die Benutzung der Eigenschaftlisten stehen erst einmal folgende drei Befehle zur Verfügung:

1. Einem Objekt (Namen) wird eine Eigenschaftsbindung zugewiesen:

- SETZEEIGENSCHAFT (kurz: SEIG)

2. Eine Eigenschaft zu einem Objekt (Namen) und einer Eigenschaftsbezeichnung wird abgefragt:

 - HOLEEIGENSCHAFT (kurz: HEIG)

3. Eine Eigenschaftsbindung wird für ein Objekt (Namen) wieder gelöscht:

 - VERGISSEIGENSCHAFT (kurz: VEIG)

 Ein Beispiel:

Es soll eine Eigenschaftsliste zu dem Objekt BAUM erstellt werden. In dieser Liste soll für die Eigenschaftsbezeichnung "GROESSE" der Wert 10 stehen:

```
?SETZEEIGENSCHAFT "BAUM "GROESSE 10
?HOLEEIGENSCHAFT "BAUM "GROESSE
 Ergebnis = 10
?VERGISSEIGENSCHAFT BAUM "GROESSE
?HOLEEIGENSCHAFT "BAUM "GROESSE
 Zu : GROESSE keine Eigenschaft bekannt unter : BAUM
?...
```

Es wurde als erstes die Eigenschaft zugewiesen. Dann wurde die Eigenschaft wieder abgefragt - das Ergebnis war 10. Als drittes wurde die Eigenschaft wieder gelöscht. Dies führt bei erneuter Abfrage der Eigenschaft zu einer Fehlermeldung, da die Eigenschaft ja nicht mehr existiert.

Wir wollen nun einem Objekt - hier einer Person - einige Eigenschaften zuweisen: Die Person heißt RALF. Die Eigenschaften von RALF sind:

```
HAARE     = BLOND
AUGEN     = BRAUN
BRILLE    = FALSCH
GROESSE   = 180
GEWICHT   = 80
WOHNUNG   = (2000 HAMBURG)
```

Diese Eigenschaften wollen wir nun eingeben. Dazu benutzen wir die Abkürzung SEIG für SETZEEIGENSCHAFT.

```
?SEIG "RALF "HAARE "BLOND
?SEIG "RALF "AUGEN "BRAUN
?SEIG "RALF "BRILLE FALSCH
?SEIG "RALF "GROESSE 180
?SEIG "RALF "GEWICHT 80
?SEIG "RALF "WOHNUNG [2000 HAMBURG]
?...
```

Wir können diese Eigenschaften von RALF nun im einzelnen abfragen (HEIG ist die Kurzform von HOLEEIGENSCHAFT!):

```
?HEIG "RALF "BRILLE
 Ergebnis = FALSCH
?DZ HEIG "RALF "HAARE
BLOND
?DZ HEIG "RALF "GEWICHT
80
?DZ HEIG "RALF "WOHNUNG
2000 HAMBURG
?...
```

Wir können die gespeicherten Eigenschaften auch wieder überschreiben. Dazu brauchen wir nur erneut eine Eigenschaft mit dem Befehl SETZEEIGENSCHAFT zuzuweisen - die alte gespeicherte Eigenschaft wird dann automatisch vergessen (gelöscht):

```
?DZ HEIG "RALF "HAARE
BLOND
?SEIG "RALF "HAARE "ROT
?DZ HEIG "RALF "HAARE
ROT
?...
```

Nun sieht die von AIP - LOGO gespeicherte Eigenschaftsliste zum Objekt (Namen) RALF folgendermaßen aus:

Eigenschaftsliste zu RALF:

```
(RALF   (WOHNUNG (2000 HAMBURG))  (GEWICHT 80)  (GROESSE 180)
        (BRILLE FLASCH)  (AUGEN BRAUN)  (HAARE ROT))
```

Zur besseren Übersicht ist die Eigenschaftsliste zu RALF in zwei Zeilen geschrieben worden.

Jetzt soll die Eigenschaft GEWICHT wieder vergessen (gelöscht) werden:

```
?VERGISSEIGENSCHAFT "RALF "GEWICHT
?...
```

Danach sieht unsere Eigenschaftsliste zu RALF folgendermaßen aus:

```
(RALF   (WOHNUNG (2000 HAMBURG))  (GROESSE 180)  (BRILLE FALSCH)
        (AUGEN BRAUN)  (HAARE ROT))
```

Es können beliebig viele Eigenschaftslisten zu verschiedenen Objekten erstellt und bearbeitet werden. Wir können aber auch mit den Eigenschaftslisten als Ganzes arbeiten. Dazu sind die folgenden drei Befehle zu benutzen:

- SETZEEIGENSCHAFTSLISTE (kurz: SEIGL)
- HOLEEIGENSCHAFTSLISTE (kurz: HEIGL)
- VERGISSEIGENSCHAFTSLISTE (kurz: VEIGL)

Die noch zu RALF gespeicherte Eigenschaftsliste können wir uns damit vollständig ausgeben lassen:

```
?HOLEEIGENSCHAFTSLISTE "RALF
 Ergebnis = (RALF (WOHNUNG (2000 HAMBURG)) (GROESSE 180) (BRILLE FALSCH)
(AUGEN BRAUN) (HAARE ROT))
?...
```

Wir können auch für die Person CLAUDIA eine Eigenschaftsliste definieren. Folgende Eigenschaften sollen enthalten sein:

```
HAARE     = SCHWARZ
AUGEN     = BLAUGRUEN
BRILLE    = WAHR
GROESSE   = 165
WOHNUNG   = (2300 KIEL)
```

Die Eigenschaftsliste muß folgendermaßen aussehen:

Eigenschaftsliste zu CLAUDIA:

```
(CLAUDIA  (WOHNUNG (2300 KIEL))  (GROESSE 165)  (BRILLE WAHR)
          (AUGEN BLAUGRUEN)  (HAARE SCHWARZ))
```

Wir können diese Eigenschaftsliste jetzt als Ganzes eingeben:

```
?SEIGL "CLAUDIA [(WOHNUNG (2300 KIEL)) (GROESSE 165) (BRILLE WAHR)
(AUGEN BLAUGRUEN) (HAARE SCHWARZ)]
?DZ HEIG "CLAUDIA "WOHNUNG
2300 KIEL
?...
```

Auf diese Weise können wir auch eine alte gespeicherte Eigenschaftsliste überschreiben. Wenn wir eine Eigenschaftsliste mit dem Befehl SETZEEIGENSCHAFT zuweisen, wird automatisch die alte Eigenschaftsliste vergessen (gelöscht).
Wir können auch die Zuweisung einer Eigenschaftsliste vollständig löschen. Nach Aufruf des Befehls VERGISSEIGENSCHAFTSLISTE wird die Zuweisung einer Eigenschaftsliste vergessen (gelöscht). Wenn dann wieder die Eigenschaftsliste oder auch nur eine einzelne Eigenschaft abgefragt wird, führt dies zu einer Fehlermeldung.

```
?VERGISSEIGENSCHAFTSLISTE "CLAUDIA
?HEIGL "CLAUDIA
 Keine Eigenschaften bekannt unter : CLAUDIA
?HEIG "CLAUDIA "GROESSE
 Keine Eigenschaften bekannt unter : CLAUDIA
?...
```

Es wird bei der Abfrage HOLEEIGENSCHAFT im Fehlerfall genau unterschieden, ob die Eigenschaft nicht existiert, oder ob es gar keine Eigenschaftsliste zu der Person (dem Objekt) gibt:

```
?HOLEEIGENSCHAFT "CALUDIA "BRILLE
 Keine Eigenschaften bekannt unter : CLAUDIA
?SETZEEIGENSCHAFT "RALF "HAARE "BLOND
?HOLEEIGENSCHAFT "RALF "BLABLABLA
 Zu : BLABLABLA keine Eigenschaft bekannt unter : RALF
?...
```

Es gibt in AIP - LOGO jetzt auch noch zwei Fragen nach der Existenz von Eigenschaften:

- EIGENSCHAFT? (kurz: EIG?)
- EIGENSCHAFTSLISTE? (kurz: EIGL?)

Zur Erläuterung der Benutzung dieser beiden Fragen zwei Beispiele:

```
?SETZEEIGENSCHAFT "RALF "HAARE "BLOND
?EIGENSCHAFT? "RALF "HAARE
 Ergebnis = WAHR
?EIGENSCHAFT? "RALF "TELEFON
 Ergebnis = FALSCH
?EIGENSCHAFTSLISTE? "RALF
 Ergebnis = WAHR
?EIGENSCHAFTSLISTE? "INGA
 Ergebnis = FALSCH
?...
```

So kann geprüft werden, ob der Aufruf der Befehle HOLEEIGENSCHAFT bzw. HOLEEIGENSCHAFTSLISTE - mit den gleichen Eingaben - zu einer Fehlermeldung führen würden oder nicht.

Wie schon in Kapitel 5 beschrieben, können alle Eigenschaften mit

- VERGISS EIGENSCHAFTEN (VG EIGS)

gelöscht werden. Mit

- BEWAHRE EIGENSCHAFTEN (BW EIGS)

werden sie auf Cassette gespeichert (auch bei BEWAHRE ALLES !).

- ZEIGE EIGENSCHAFTEN (ZG EIGS)

Hierbei werden alle Eigenschaften, die im Arbeitsspeicher stehen, ausgelistet. Dabei werden alle Eigenschaften - jeweils nach dem Objekt (dem Namen bzw. der Person) sortiert - untereinander aufgeführt. Dieses verschafft Ihnen einen guten Überblick über die schon bekannten Eigenschaftsbindungen.

9.6 Allgemeine Eigenschaftslisten

In AIP - LOGO werden noch zwei Befehle für die Eigenschaftslistenverarbeitung zur Verfügung gestellt. Diese beziehen sich aber nicht auf die gespeicherten Eigenschaftslisten, sondern immer auf eine als Eingabe mitgegebene Liste.

- SUCHE
- TEST?

Der Befehl SUCHE und die Frage TEST? brauchen zwei Eingaben. Als erstes das in der Eigenschaftsliste zu suchende Objekt, und als zweites eben diese Eigenschaftsliste selbst.

Dazu ein Beispiel:

```
?SETZE "ELISTE [JOCHEN (HAARE BLOND) (AUGEN BRAUN) (TELEFON 333222)]
?SUCHE "AUGEN :ELISTE
 Ergebnis = (AUGEN BRAUN)
?SUCHE "WOHNUNG :ELISTE
 Ergebnis = ()
?TEST? "AUGEN :ELISTE
 Ergebnis = WAHR
?TEST? "WOHNUNG :ELISTE
 Ergebnis = FALSCH
?...
```

Der Befehl SUCHE sucht nach einer Unterliste der - als zweite Eingabe gegebenen - Eigenschaftsliste, deren erstes Element gleich der ersten Eingabe ist. Wenn mehrere solche Unterlisten vorhanden sind, deren erstes Element gleich der ersten Eingabe ist, so wird die erste gefundene Unterliste zurückgegeben. Wenn keine Unterliste gefunden wird, wird die leere Liste zurückgegeben (das vermeidet Fehlermeldungen!).
Die Frage TEST? fragt nun danach, ob es eine solche Unterliste gibt. Es wird dann WAHR zurückgegeben, wenn es in der gegebenen Liste diese Unterliste gibt.

Der Befehl SUCHE ist sehr nützlich bei der Aufstellung von Programmen, die mit "Wissen" arbeiten müssen. Es wird hier eine Möglichkeit eröffnet, im Computer Wissen darzustellen - als eine Eigenschaftsliste. Dieser Befehl ist weitaus vielseitiger, als die Befehle zu den intern gespeicherten Eigenschaftslisten.

10. AIP - LOGO "INTERN"

10.1 Interner Aufbau von Objekten in AIP - LOGO

Dieses Kapitel ist denen gewidmet, die sich näher mit dem Aufbau listenorientierter Programmiersprachen oder dem Gebiet der künstlichen Intelligenz beschäftigen wollen.
Zum Verständnis wird tlw. weiteres Wissen über Maschinensprache und Computer generell vorausgesetzt. Als Programmier-Anfänger sollten Sie dieses Kapitel erst einmal übergehen.

Von außen aus sehen wir von den Listen nur die Klammern. Intern sind sogenannte "Knoten" für die Reihenfolge der Elemente verantwortlich. Alle Knoten nehmen im Speicher 5 Bytes ein. Jeder Knoten steht an einer bestimmten Stelle des Speichers. Wir wollen die Anfangsadresse dieses fünf-byte Blocks als Adresse des Knotens betrachten. Diese Adresse ist ein zwei-Byte bzw. 16-Bit-Wert.

```
+----+----+----+----+----+
!Code! A1 ! A2 ! B1 ! B2 !          Knoten in LOGO
+----+----+----+----+----+
  !
  Adresse des Knotens
```

Der ganze verfügbare Speicher in AIP - LOGO besteht nun also ausschließlich aus Knoten. Alle Objekte in AIP - LOGO werden durch solche Knoten ausgedrückt. Dies ist eigentlich nicht üblich in listenorientierten Programmiersprachen, aber der beschränkte Speicherplatz, der mit dem Z-80 Mikroprozessor ansprechbar ist, macht diese Technik hier sinnvoll.

Zum Aufbau von Knoten:

Das erste Byte eines jeden Knotens ist der Code. Dieser Code sagt aus, um was für ein Objekt es sich handelt.

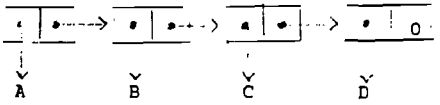
CODE	OBJEKT
00h	Liste
01h	Atom (Wort)
02h	Kommentar
03h	Realzahl
04h	Integerzahl
FFh	dieser Knoten ist nicht benutzt

Der Knoten einer Liste (Code = 00h) soll zwei Aussagen haben. Erstens, welches Objekt das erste Element dieser Liste ist, und zweitens, welche Adresse die Restliste (ohne das erste Element) hat. Eine Liste benötigt genau so viele Knoten, wie sie Elemente hat.

10.1.1 Die Zeigerdarstellung

Man kann die Knoten von Listen gut als ein aus zwei Quadraten gebautes Rechteck darstellen. Von jedem Quadrat geht ein Zeiger zu einem anderen Knoten aus.

(A B C D) =



Die Null im letzten Quadrat bedeutet, daß hier auf keinen weiteren Knoten verwiesen wird. Die Liste ist hier zu Ende. Dies wird durch die Adresse 0000h beschrieben. Wir wollen diese Adresse als NIL (= nichts) bezeichnen. Ebenso wollen wir das erste Element einer Liste immer als CAR (sprich: kar) der Liste und die Liste ohne ihr erstes Element als CDR (sprich: kudder) der Liste bezeichnen. Diese Formulierungen stammen von der Sparche LISP ab und sind gut geeignet, auch die Arbeitsweise von AIP - LOGO zu erklären.

```
Liste = (A B C D)
CAR   = A
CDR   = (B C D)

NIL   = () = leere Liste
```

Die Adresse des CAR eines Listen-Knotens ergibt sich aus den Bytes A1 und A2. Die höherwertigen 8 Bits stehen in A2. Ebenso ergibt sich der CDR durch die Bytes B1 und B2. Wenn die Liste nur ein Element hat, also der CDR = NIL ist, sind die Bytes B1 und B2 beide null.
Aus diesem Aufbau der Listen in AIP - LOGO wird einsichtig, warum die Befehle ERSTES und OHNEERSTES keinen großen Arbeitsaufwand benötigen. Es wird einfach der CAR oder CDR zurückgegeben.
Alle Daten werden in AIP - LOGO durch ihre Adressen übergeben. Um also das erste Element einer Liste zu ermitteln, ist es nur nötig, den Inhalt von (Add+1) und (Add+2) zu einer 16-Bit Adresse zu kombinieren.

Nun wollen wir uns dem Aufbau der anderen Objekte in AIP - LOGO zuwenden.

- Worte

(Code = 01h) werden in LOGO sehr kompliziert dargestellt. Der erste Knoten, der zum Wort gehört, enthält nur zwei Buchstaben und die Adresse, wo das Wort weitergeht. Diese weiteren Knoten haben keinen Code. Stattdessen sind die ersten drei Bytes (Code, A1 und A2) die Buchstaben und die beiden restlichen (B1 und B2) die Adresse, wo das Wort weitergeht. Den Schluß des Wortes markiert das Byte 0Dh, und die Bytes B1 und B2 des letzten Wortknotens sind null. Das Wort "LOGO" benötigt zwei Knoten (10 Bytes!).

Das Wort "LOGO" ist intern:

	Code	A1	A2	B1	B2
1. Knoten	01h	L	O	Add	Add
2. Knoten	G	O	ODh	0	0

Obwohl der zweite und jeder weitere Knoten von einem Wort als Code keinen konstanten Wert haben, sind sie von allen anderen klar zu unterscheiden. Kein Zeichen oder Buchstabe kann einen Wert von 00h, 01h, 02h, 03h, 04h oder FFh annehmen.

Achtung: Interne Worte - die Grundworte in AIP - LOGO - stehen in einem anderen Teil des Speichers geschrieben. Sie bestehen aus dem Code 01h gefolgt von allen Buchstaben und dem Schluß 0Dh in einer Kette. Zur Unterscheidung ist jeweils beim ersten Buchstaben das 7. Bit (höchstes Bit) zusätzlich gesetzt.

- Kommentare unterscheiden sich von Worten nur dadurch, daß auch Leerstellen und Klammern in ihnen auftauchen können. Sie werden intern nicht weiter bearbeitet, sondern können nur ein- und ausgegeben werden.

- Realzahlen sind in zwei Knoten untergebracht. Es handelt sich um 5 Byte-Realzahlen. Zwei Bytes werden in dem ersten Knoten und die drei anderen Bytes in dem zweiten Knoten untergebracht. Der Code ist von beiden Knoten 03h. Im ersten Knoten wird in B1 und B2 noch die Adresse des zweiten Knotens mitgeteilt. Das Byte B2 des zweiten Knotens wird nicht genutzt.
Dabei bestehen die Realzahlen aus einer vierbyte-Mantisse und im ersten der fünf Bytes steht der Exponent und das Vorzeichen.

- Integerzahlen bestehen aus nur vier Bytes und können in einem Knoten untergebracht werden.

10.2 Die maschinennahen Befehle

AIP - LOGO stellt einige Befehle zur Verfügung, durch die Verbindung zu Maschinenprogrammen erstellt werden kann. Es ist aber nicht möglich, auf die Sytemroutinen des ROM's zuzugreifen, da auch AIP - LOGO diese nicht benutzt. Sie können aber mit dem Befehl

- ADE

AIP - LOGO verlassen und in das Monitor-ROM Ihres Computers springen.
AIP - LOGO kann dann aber nicht wieder gestartet werden. Es wird dabei leider (zumindest teilweise) zerstört.

AIP - LOGO stellt zwei Umrechnungsbefehle zur Verfügung:

- ZAHL$
- HEX$

Dazu einige Beispiele:

```
?DZ HEX$ 1000
03E8
?HEX$ 65534
 Ergebnis = FFFE
?ZAHL$ "H3E9
 Ergebnis = 1001
?DZ ZAHL$ "B00110101
53
?...
```

Der Befehl HEX$ wandelt eine ganze Zahl zwischen 0 und 65535 in ein Wort mit vier Zeichen (Hexadezimalziffern) um. Dieses Wort ist die entsprechende Zahl in hexadezimaler Darstellung. Das Wort kann - wie gewohnt - weiterverarbeitet werden.
Der Befehl ZAHL$ wandelt ein Wort in eine Dezimalzahl um. Dabei muß das erste Zeichen ein "H" für hexadezimal - oder ein "B" für binär sein. Das Wort muß dann danach entweder eine maximal achtstellige Hexadezimalzahl oder aber eine maximal 32-stellige Binärzahl darstellen. Dieses Wort wird dann in eine Intergerzahl umgewandelt, die auch wie gewohnt weiterverarbeitet werden kann.

Mit den folgenden Befehlen können einzelne Bytes an einer bestimmten Speicheradresse abgelegt bzw. herausgeholt werden (vergleichbar mit den BASIC-befehlen POKE und PEEK!).

- LEGE
- HOLE

Der Befehl LEGE braucht zwei Eingaben. Die erste ist die Adresse (0 bis 65535), wo das Byte abgelegt werden soll, die zweite Eingabe enthält das Byte (0 bis 255).
Der Befehl HOLE braucht nur die Adresse (0 bis 65535) als Eingabe und gibt den Wert des dort im Speicher stehenden Bytes als Integerzahl zurück.

Dazu zwei Beispiele:

```
?LEGE 62000 34
?HOLE 62000
 Ergebnis = 34
?...
```

A C H T U N G :

Wenn Sie innerhalb des Speichers, der von AIP - LOGO belegt oder genutzt wird, irgend etwas "hineinlegen", können Sie damit das Programm sehr schnell zerstören. Um dies zu vermeiden, und trotzdem einen Bereich für derartige Speicheroperationen zur Verfügung zu haben, können Sie sich mit dem Befehl LIMIT von AIP - LOGO einen Speicherbereich für diese Zwecke zur Verfügung stellen lassen.

- LIMIT

Eines muß hierbei beachtet werden: Der Befehl LIMIT muß sofort nach Start von AIP - LOGO ausgeführt werden, da der betreffende Speicherbereich sonst eventuell schon von AIP - LOGO genutzt wird.
Der Speicherbereich, der zur Verfügung gestellt werden kann, geht grundsätzlich von der zu dem Befehl LIMIT eingegebenen Adresse bis zum Anfang des "Stackbuffers" (Stapelbuffer) von AIP - LOGO (die speziellen Adressen finden Sie im Anhang Teil I.).

Es ist in AIP - LOGO auch möglich, ein Maschinenprogramm aufzurufen:

- RUFE

Dieser Befehl ruft ein in Maschinensprache geschriebenes Unterprogramm auf, das an der als Eingabe gegebenen Adresse gestartet wird. Dabei brauchen die Register des Prozessors nicht "gerettet" zu werden!

Da es sehr mühselig ist, ein Maschinenprogramm so in Hexzahlen einzugeben, besteht in AIP - LOGO auch die Möglichkeit, ein Maschinenprogramm oder auch Maschinendaten über Cassette / Diskette einzuladen. Dazu wird der Befehl

- LADE$

verwendet. Er wird wie der Befehl LADE mit einem Dateinamen aufgerufen (siehe dazu Kapitel 5). Das Maschinenprogramm bzw. die Maschinendaten müssen in einem vorher durch LIMIT zur Verfügung gestellten Bereich liegen. Ansonsten wird eine Fehlermeldung gegeben. Das Maschinenprogramm kann z.B. vorher auf einen Assembler - natürlich in den betreffenden Bereich - geschrieben und "abgesaved" werden. Es kann dann von AIP - LOGO aus eingeladen werden.

10.3 Destuktive Listenverarbeitung

Wenn wir Listen verarbeiten, erstellen wir dabei immer neue Listen, die unter Umständen Teile der alten Listen enthalten. Wir können Listen aber auch verändern, indem wir die Knoten der Listen manipulieren. Der Nachteil dieses "destruktiven" Verfahrens ist, daß sich zum einen Werte von Namen oder Körper von Prozeduren verändern, obwohl wir dies gar nicht wollen. Außerdem ist es möglich, unendliche Listen (in sich geschlossene Listen) zu schaffen, die zum Absturz des Systems führen können.

Der große Vorteil ist, daß dieses Verfahren erheblich schneller arbeitet, als übliche Methoden. Auch besteht hier die Möglichkeit, dieses Verfahren für die KI (s. Kap. 9) zu nutzen.

AIP - LOGO stellt zwei Befehle zur Verfügung, die den übergang von LOGO - Daten in maschinennahe Daten ermöglichen.

- KNOTEN
- OBJEKT

Der Befehl KNOTEN liefert die Startadresse des Knotens der Eingabe.

Die Eingabe wird als Adresse für einen Knoten gedeutet. Das Objekt, das dieser Knoten ausdrückt, wird zurückgegeben.

Ein Beispiel:

```
?SETZE "A "MASCHINE
?DZ KNOTEN :A
35402                          ;Diese Adresse ist natürlich immer verschieden.
?OBJEKT 35402
 Ergebnis = MASCHINE
?...
```

Es werden jetzt zwei Prozeduren vorgestellt, die den CAR bzw. den CDR einer Liste durch ein neues Objekt "ersetzen". Dabei wird die Quelle dieser Liste verändert. Dies bedeutet, daß sich diese Liste überall, wo sie - evt. als Unterliste - auftaucht, im Speicher geändert hat.

Nun die beiden Prozeduren:

```
PR ERSETZECAR :NEUERCAR :LISTE
 LEGE (KNOTEN :LISTE) + 1 (REST KNOTEN :NEUERCAR 256)
 LEGE (KNOTEN :LISTE) + 2 (DIV KNOTEN :NEUERCAR 256)
ENDE

PR ERSETZECDR :NEUERCDR :LISTE
 LEGE (KNOTEN :LISTE) + 3 (REST KNOTEN :NEUERCDR 256)
 LEGE (KNOTEN :LISTE) + 2 (DIV KNOTEN :NEUERCDR 256)
ENDE
```

Durch diese Prozeduren entstehen - wie gesagt - auch an anderer Stelle noch Änderungen. Dazu ein Beispiel:

```
?SETZE "A [1 2 3 4]
?ERSETZECAR [A B C] :A
?DZ :A
(A B C) 2 3 4
?ERSETZECAR "LOGO 0E 0E :A
?:A
((A B C) 2 LOGO 4)
```

Um ein Fazit dieser Methode zu ziehen: Es ist sehr praktisch, auf diese Weise Listen zu manipulieren. Aber es muß dabei äußerst vorsichtig vorgegangen werden, damit das System nicht abstürzt, oder man nicht gewünschte Daten zerstört.

A N H A N G :

I. Bildschirm und Besonderheiten

AIP - LOGO läuft allgemein im 40-Zeichenmodus. Da der MZ-800 aber auch im 80-Zeichenmodus betrieben werden kann, stell AIP - LOGO den Befehl

- MODUS (kurz: MOD)

zur Verfügung.

MODUS 40 schaltet in den 40-Zeichenmodus und
MODUS 80 schaltet in den 80-Zeichenmodus.

So können z.B. für den Befehl BLINKER (s. Kap. 7.4) im 40-Zeichenmodus die Spalten 0 - 39 und im 80-Zeichenmodus die Spalten 0 - 79 angesprungen werden. Da der MZ-800 25 Zeilen auf dem Bildschirm hat, können Zeilen von 0 - 24 angesprungen werden. (Im BILD-Modus allerdings nur die untersten fünf: 20 - 24.)

Tabelle A.1 zeigt die Farbcodes für die auf dem MZ-800 darstellbaren Farben. Diese Codes werden als Eingaben für die Befehle HINTERGRUND, RANDFARBE und PALETTE benötigt.

(Tabelle A.1)

Farbe für: HINTERGRUND, RANDFARBE oder PALETTE		Codezahl
	schwarz	0
	blau	1
	rot	2
	rosa	3
	grün	4
	türkis	5
	gelb	6
	weiß	7
	grau	8
helles	blau	9
helles	rot	10
helles	rosa	11
helles	grün	12
helles	türkis	13
helles	gelb	14
helles	weiß	15

Die Voreinstellung lautet: Hintergrund = 0 und Randfarbe = 0.

Die Befehle PALLETTE und PALLETTENEU können auf allen SHARP MZ-8xx benutzt werden.

Die Speicherorganisation:

AIP - LOGO stellt mit dem Befehl LIMIT dem Programmierer einen Teil des Speichers für maschinennahe Nutzung zur Verfügung.

Dieser Teil des Speichers ist beschränkt durch den Anfang des Stacks (s. Kap. 10.2) und das Ende des mindest-Hauptspeichers.

Es können als Eingaben für LIMIT nur Werte zwischen 49162 und 61183 eingegeben werden, da der Systemstack bei 61184 beginnt.

Es kann also maximal der Bereich von 49162 bis 61183 mit dem Befehl LIMIT für maschinennahe Nutzung reserviert werden.

Der ASCII-Code des Sharp MZ-800:

Code	Zeichen	Code	Zeichen		Code	Zeichen	Code	Zeichen
32	Leerst.	64			128	Leerst.	160	q
33	!	65	A		129		161	a
34	"	66	B		130		162	z
35	#	67	C		131		163	w
36	$	68	D		132		164	s
37	%	69	E		133		165	u
38	&	70	F		134		166	i
39	'	71	G		135		167	
40	(	72	H		136		168	
41	)	73	I		137		169	k
42	*	74	J		138		170	f
43	+	75	K		139		171	v
44	,	76	L		140		172	
45	-	77	M		141		173	
46	.	78	N		142		174	
47	/	79	O		143		175	j
48	0	80	P		144		176	n
49	1	81	Q		145		177	
50	2	82	R		146	e	178	
51	3	83	S		147		179	m
52	4	84	T		148		180	
53	5	85	U		149		181	
54	6	86	V		150	t	182	
55	7	87	W		151	g	183	o
56	8	88	X		152	h	184	l
57	9	89	Y		153		185	
58	:	90	Z		154	b	186	
59	;	91	[		156	x	187	
60	<	92	\		157	d	188	
61	=	93	]		158	r	189	y
62	>	94			159	p	190	
63	?	95			161	c	191	

Die leeren Felder sind nicht im normalen ASCII-Zeichensatz enthaltene Zeichen.

Der Centronics ASCII-Code (für AIP - LOGO von Bedeutung!):

Code	Zeichen	Code	Zeichen	Code	Zeichen
32	Leerst.	64		96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92		124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_	127	

Wenn Sie einen Centronics-kompatiblen Drucker angeschlossen haben, sollten Sie dies AIP - LOGO mit dem Befehl

- DRN

mitteilen. Wenn Sie einen Sharp-Drucker angeschlossen haben, so teilen Sie dies AIP - LOGO bitte mit dem Befehl

- DRS

mit. Es werden dann jeweils alle Zeichen von AIP - LOGO aus richtig auf dem Drucker ausgegeben.

II. Sondertasten

Es sollen noch einmal alle Tasten mit besonderer Bedeutung mit einer Kurzbeschreibung aufgeführt werden.

"Ctrl-A" Schaltet zwischen VERSTECKIGEL und ZEIGIGEL hin und her, auch während der Programmausführung.

"Ctrl-B" Schaltet zwischen TEILBILD und VOLLBILD hin und her, auch während der Programmausführung.

"SHIFT"+"BREAK"
"Ctrl-C" Bricht die Programmausführung vollständig ab.

"Ctrl-P" Schaltet die Ausgabe auf dem Drucker ein und aus. Ist auch während der Programmausführung möglich.

"Ctrl-Q" Schaltet das Ablaufprotokoll ein und aus, auch während der Programmausführung.

"BREAK"
"Ctrl-S" Hält das laufende Programm an und wartet auf einen Tastendruck.

leere Taste über "CR"
"Ctrl-Z" Unterbricht das laufende Programm und springt in den Unterbrechungsmodus.

"TAB" entspricht dem Anführungszeichen ("SHIFT" + "2")

III. Kurzbeschreibung Editor

Der AIP - LOGO Editor kann mit den folgenden Befehlen aufgerufen werden:

LERNE PROZNAME	Eine noch nicht bestehende Prozedur soll gelernt werden.
EDIT PROZNAME	Eine schon bestehende Prozedur soll verändert bzw. berichtigt werden.
EDIT (ohne Namen)	Die zuletzt bearbeitete Prozedur soll wieder bearbeitet werden. Es wird auch gleich in die zuletzt in dieser Prozedur bearbeitete Zeile gesprungen.
ED (mit/ohne Namen)	Genauso wie EDIT zu benutzen.
EDF (ohne Namen)	Die Prozedur, inder zuletzt von AIP - LOGO ein Fehler gemeldet wurde, soll bearbeitet bzw. berichtigt werden. Es wird auch gleich an die fehlerhafte Zeile gesprungen.

Nun noch einmal alle Befehle im AIP - LOGO Editor auf einen Blick mit einer Kurzbeschreibung:

Kurzbeschreibung AIP - LOGO Editor: (SHARP MZ-800)

"Pfeil runter" oder "CR" = RETURN	eine Zeile weiter	+ 1
"Pfeil hoch"	eine Zeile zurück	- 1
"Ctrl-T"	Springen auf die Titelzeile	(Titelzeile)
leere Taste über "CR", "Ctrl-Z"	Zeigen der Prozedur	
"Ctrl-A"	Springen auf die erste Zeile	(Anfang)
"Ctrl-E"	Springen auf die letzte Zeile	(Ende)
"Ctrl-I"	Einfügen einer Zeile vor der momentanen Zeile	
"Ctrl-D"	Löschen der momentanen Zeile	
"Ctrl-N"	Neustarten der Berichtigung	
"GRAPH" oder "Ctrl-L"	Beenden der Berichtigung und Lernen der Prozedur	
"Shift-Break" oder "Ctrl-C"	Abbrechen der Berichtigung, alte Prozedur bleibt erhalten	
"Pfeil rechts", "Pfeil links" und "DEL" , "INST"	können wir gewohnt benutzt werden!	

IV. Befehlsliste mit Abkürzungen

Es werden noch einmal alle Grundwörter von AIP - LOGO mit ihren Abkürzungen und dem Kapitel, in dem sie beschrieben sind, aufgelistet
Sie stehen in alphabetischer Reihenfolge:

Seite	AIP - LOGO Grundwort	Abkürzung	Kapitel
124	ADE		10.2
65	ALLES	AL	5.1
49	ALLE?		3.3.2
76	ARCTAN		6.1.3
97	ASC		7.4.1
53	AUFKURS	AK	3.4
52	AUFX		3.4
52	AUFXY		3.4
52	AUFY		3.4
107	AUSSTIEG	AS	8.3
109	BETRIEB	BTR	8.4
68	BEWAHRE	BW	5.4.1
71	BEWAHREBILD	BWB	5.5
97	BLINKER		7.4
16	BILD	BI	1.4
76	COS		6.1.3
47	DANN		3.3.1
116	DEF		9.4
75	DIV		6.1.1
132	DRN		Anhang I
132	DRS		Anhang I
96	DRUCK		7.3
96	DRUCKE	DR	7.3
98	DRUCKERAUS	DA (Ctrl-P)	7.5.2
98	DRUCKEREIN	DE (Ctrl-P)	7.5.2
12, 95	DRUCKEZEILE	DZ	1.2, 7.3
30, 108	EDF		2.3.1, 8.3.1
28	EDIT	ED	2.3
65	EIGENSCHAFTEN	EIGS	5.1
120	EIGENSCHAFT?	EIG?	9.5
120	EIGENSCHAFTSLISTE?	EIGL?	9.5
49	EINES?		3.3.2
93	EINGABE	EG	7.2.1
90	ELEMENT?	EL?	6.4.7
24	ENDE		2
54	ENTFERNUNG	EF	3.4
78, 81	ERSTES	ER	6.2, 6.3.2
77	EXP		6.1.3
48	FALSCH		3.3.1
17	FARBE		1.4.2
72	FDD		5.6
101	FTASTE		7.6

Seite	AIP - LOGO Grundwort	Abkürzung	Kapitel
101	FTASTEN		7.6
58	FUELLE		3.5
67	GCOLL		5.3
91	GEHE		7.1.1
88	GLAETTE		6.4.5
107	HALT		8.3
124	HEX$		10.2
55	HINTERGRUND	HG	3.5
125	HOLE		10.2
117	HOLEEIGENSCHAFT	HEIG	9.5
118	HOLEEIGENSCHAFTSLISTE	HEIGL	9.5
60	IGELGROESSE	IGR	3.6
58	IGELZUSTAND	IZ	3.5
72	INHALT	IH	5.6
73	INT		6.1
57	INVERSAUS	IA	3.5
57	INVERSEIN	IE	3.5
74	IZAHL?		6.1.1
85	KLAMMER		6.4.1
99	KNOPF?		7.5.3
126	KNOTEN		10.3
54	KURS		3.4
70	LADE		5.4.2
71	LADEBILD	LDB	5.5
70, 126	LADE$		5.4.2, 10.2
86	LAENGE		6.4.2
125	LEGE		10.2
24	LERNE	PR	2.1
78, 81	LETZTES	LZ	6.2, 6.3.2
125	LIMIT		10.2
16	LINKS	LI	1.4
82	LISTE		6.3.2
89	LISTEOHNE	LO	6.4.6
89	LISTEOHNEALLE	LOA	6.4.6
83	LISTE?		6.3.2
77	LN		6.1.3
38	LOESCHEBILD	LB	3
16	LOESCHESCHIRM	LS	1.4
97	LOESCHETEXT	LT	7.4
77	LOG		6.3.1
58	MALE		3.5
82	MITERSTEM	ME	6.3.2
82	MITLETZTEM	ML	6.3.2
38	MITTE		3
129	MODUS	MOD	Anhang I
64	NAME?		4
65	NAMEN		5.1
49	NICHT		3.3.2

Seite	AIP - LOGO Grundwort	Abkürzung	Kapitel
126	OBJEKT		10.3
78, 81	OHNEERSTES	OE	6.2, 6.3.2
78, 81	OHNELETZTES	OL	6.2, 6.3.2
56	PALETTE	PAL	3.5
56	PALETTENEU	PALN	3.5
107	PAUSE		8.3
87	PICKE		6.4.3
115	PRLISTE		9.4
109	PROTOKOLLAUS	PA (Ctrl-Q)	8.4
109	PROTOKOLLEIN	PE (Ctrl-Q)	8.4
65	PROZEDUREN	PR	5.1
48	PROZEDUR?		3.3.1
55	RAND		3.5
56	RANDFARBE	RF	3.5
55	RANDSPRUNG	RS	3.5
16	RECHTS	RE	1.4
75	REST		6.1.1
53	RICHTUNG	RI	3.4
126	RUFE		10.2
73	RUNDE		6.1
35	RUECKGABE	RG	2.7
36	RUECKKEHR	RK	2.7
16	RUECKWAERTS	RW	1.4
82	SATZ		6.3.2
62	SETZE		4
116	SETZEEIGENSCHAFT	SEIG	9.5
118	SETZEEIGENSCHAFTSLISTE	SEIGL	9.5
76	SIN		6.1.3
60	SKALA		3.6
47	SONST		3.3.1
77	STARTEZUFALL	SZ	6.1.3
38	STARTIGEL	SI	3
99	STEUER		7.5.3
17	STIFTAB	SA	1.4.2
17	STIFTHOCH	SH	1.4.2
121	SUCHE		9.6
76	TAN		6.1.3
94	TASTE		7.2.2
95	TASTEO		7.2.2
94	TASTE?		7.2.2
17	TEILBILD	TB (Ctrl-B)	1.4.2
50	TESTE		3.3.3
121	TEST?		9.6
98	TON		7.5.1
115	TUE		9.3
92	TUEBIS		7.1.3
92	TUESOLANGE		7.1.3
79, 88	UMKEHR		6.2, 6.4.4

Seite	AIP - LOGO Grundwort	Abkürzung	Kapitel
66	VERGISS	VG	5.2
72	VERGISSBILD	VGB	5.6
72	VERGISSDATEI	VGD	5.6
117	VERGISSEIGENSCHAFT	VEIG	9.5
118	VERGISSEIGENSCHAFTSLISTE	VEIGL	9.5
17	VERSTECKIGEL	VI (Ctrl-A)	1.4.2
17	VOLLBILD	VB (Ctrl-B)	1.4.2
16	VORWAERTS	VW	1.4
48	WAHR		3.3.1
108	WARTE		8.3
106	WEITER	WT	8.3
47	WENN		3.3.1
50	WENNFALSCH	WF	3.3.3
50	WENNWAHR	WW	3.3.3
21	WIEDERHOLE	WH	1.7
62	WERT		4
79	WORT		6.2
79	WORT?		6.2
77	WURZEL	QW	6.1.3
53	XKO		3.4
53	YKO		3.4
49, 74	ZAHL?		3.3.2, 6.1
124	ZAHL$		10.2
97	ZEICHEN		7.4.1
34, 65	ZEIGE	ZG	2.6, 5.1
17	ZEIGIGEL	ZI (Ctrl-A)	1.4.2
66	ZEIGE TITEL	ZT	5.1
77	ZUFALLSZAHL	ZZ	6.1.3
12	+		1.2
12	-		1.2
12	*		1.2
12	/		1.2
12	**		1.2
48	=		3.3.1
48	<		3.3.1
48	>		3.3.1
48	<=,=<		3.3.1
48	>=,=>		3.3.1
48	<>,><		3.3.1
13	"		1.2
27	;		2.2

V. Fehlermeldungen

Die Fehlermeldungen von AIP - LOGO sind allgemein sehr ausführlich und sprechen für sich.
Es werden hier jetzt die meisten Fehlermeldungen aufgeführt und kurz beschrieben.

Igel ist im Aus !

Es wurde versucht, mit dem Igel über den Rand des Bildschirmes zu springen. Dies ist nur bei RANDSPRUNG erlaubt.

VORWAERTS ist nur im BILD-Modus erlaubt !

Es wurde versucht, einen Igel-Befehl aufzurufen, ohne vorher in den BILD-modus zu gehen.

Punkt befindet sich außerhalb !

Ein Punkt, der für einen Befehl eingegeben wurde, liegt nicht innerhalb des Igel-Koordinatenfeldes.

Zahl ist nicht in Ordnung : 3.344D23

Die Zahl entspricht nicht der geforderten Schreibweise.

* braucht Zahl davor !

Das Rechensymbol hat keine Zahl sondern ein Wort oder eine Liste als Eingabe d a v o r bekommen!

SIN braucht Zahl dahinter !

Das Vergleichssymbol hat keine Zahl sondern etwas anderes als Eingabe dahinter bekommen.

Ganze Zahl erwartet von : WIEDERHOLE

Einem LOGO-Befehl wurde keine ganze Zahl als Eingabe gegeben.

Unsinnige WIEDERHOLE-Zeile : 453

Es wurde für den Befehl WIEDERHOLE nicht - wie für die zweite Eingabe gefordert - eine Befehlszeile (-liste) eingegeben sondern eine Zahl.

Zahl zu gross/klein fuer : FARBE

Die Zahl, die als Eingabe gegeben wurde, liegt nicht in dem für diesen Befehl zugelassenen Bereich.

Ergebnis zu gross/klein von : EXP

Das Ergebnis einer Rechenoperation (-funktion) liegt außerhalb des für die Zahlen erlaubten Bereiches.

Unsinniger Dateiname : 123

Ein Dateiname muß immer mit einem großen Buchstaben beginnen.

Zu langes Wort (max. 254 Buchstaben) !

Es wurde ein zu langes Wort eingegeben oder berechnet.

WAHR/FALSCH erwartet von : TESTE

Einem LOGO-Befehl wurde kein Wahrheitswert als Eingabe gegeben.

DANN fehlt !

Es wurde WENN aufgerufen, ohne daß hinter der Bedingung ein DANN steht.

Name :BLABLABLA nicht bekannt !

Ein Name wird aufgerufen, der weder lokal noch global bekannt ist.

Prozedur PROPROPRO nicht bekannt !

Eine Prozedur wird aufgerufen, die nicht bekannt ist.

INHALT nicht möglich !

Der aufgerufene Befehl ist in dieser Version von AIP - LOGO nicht vorgesehen (z.B. ein Diskettenbefehl in einer Cassettenversion).

Merker nicht bekannt : PUNKT:

Ein Merker ist in der Prozedur nicht vorhanden.

Zu : FARBE keine Eigenschaft bekannt unter : RALF

Für die Person (das Objekt) RALF ist keine Eigenschaft mit der Eigenschaftsbeschreibung FARBE bekannt.

Keine Eigenschaften bekannt unter : PETRA

Es ist keine Eigenschaftsliste für die Person (das Objekt) PETRA bekannt.

Keine Rueckgabe von : BILD

Es wurde der Befehl BILD als Eingabe zu einem Befehl oder einer Prozedur gegeben. BILD liefert aber kein Ergebnis.

Was soll geschehen mit : 123

Es wurde ein Ergebnis berechnet bzw. eine Eingabe gegeben, ohne die Verwendung mitzuteilen.

Zu wenig Eingaben fuer : MALE

Für einen Befehl oder eine Prozedur wurden nicht genug Eingaben gegeben.

Zu wenig Eingaben in (...) fuer : MALE

Für einen Befehl oder eine Prozedur wurden innerhalb von runden Klammern nicht genug Eingaben gegeben.

Zu viel in (...) : (VORWAERTS 50 30)

Für einen Befehl oder eine Prozedur wurden innerhalb von runden Klammern zu viele Eingaben gegeben.

DANN am falschen Platz !

Ein LOGO-Befehl steht an einem nicht sinnvollen Platz. Z.B. ein DANN ohne ein WENN davor in der gleichen Zeile.

Bei Macros nur einen Namen fuer die Eingabe : ::A :B

Es wurden für eine macro Prozedur mehr als ein Name für die lokalen Namen in die Titelzeile geschrieben.

Unsinnige Eingabenforderung : 34BLA
fuer : PRONAME

In der Titelzeile einer Prozedur (hier: PRONAME) steht eine unsinnige Bezeichnung für den zu besetzenden lokalen Namen.

Unsinnige Titelzeile !

Meldung innerhalb des Editors. Die Titelzeile enthält einen unsinnigen Prozedurnamen oder unsinnige Bezeichnungen für die zu besetzenden lokalen Namen.

NAMENAME ist schon bekannt !

Es wurde versucht, mit dem Befehl LERNE eine Prozedur zu schaffen, deren Name schon bekannt ist.

BETRIEB ist LOGO-Grundwort !

Es wurde versucht, eine Prozedur zu schaffen, deren Name schon als LOGO-Grundwort bekannt ist.

Bereich ist schon benutzt, LIMIT nicht moeglich !

Der Befehl LIMIT wurde aufgerufen, nachdem von LOGO der betreffende zu limitierende Bereich bereits benutzt wurde.

Daten liegen im gesperrten Bereich !

Es wurde versucht, mit dem Befehl LADE$ eine Maschinendatei einzuladen, die in einem von LOGO benutzten Bereich liegt.

Laden ist nicht ok!

Bei einem Lade- oder Bewahrevorgang ist ein Fehler aufgetreten.

Drucker ist nicht bereit !

Der Drucker soll angesprochen werden, ist aber nicht angeschlossen oder nicht eingeschaltet.

*** Schachtelung zu tief (Stack ist voll) !

Eine Prozedur hat sich zu tief verschachtelt (max. ca. 250), dabei ist der Systemstapel zu groß geworden.

*** Speicher ist voll !

Im Speicher ist trotz Garbage-collection kein Platz mehr frei.

Unsinniger Begriff : ()

Die leere Liste ist ein für AIP - LOGO unsinniger Begriff, den es nicht auswerten kann.

")" zuviel!
"{" zuviel!
"}" zuviel!

Es wurden in einer eingegebenen Zeile mehr als eine geschweifte Klammer geöffnet oder geschlossen bzw. mehr runde (auch eckige) Klammern geschlossen als geöffnet.

"(,[,],)" nicht innerhalb von Kommentaren

Es wurde innerhalb eines Kommentars eine Klammer benutzt.

VI. Literaturhinweise

Zum Schluß dieses Handbuches möchte der Autor es nicht versäumen, noch drei Hinweise auf weiterführende Literatur zu geben. Beide Bücher haben Einfluß auf die Entwicklung von AIP - LOGO genommen.

<1> Harald Abelson: Einführung in LOGO
Übersetzt und bearbeitet von Herbert Löthe
Erschienen 1983 im IWT Verlag GmbH Vaterstetten bei München
ISBN 3-88322-023-X

<2> Christian-M. Hamann: Einführung in das Programmieren in LISP
2. Auflage
Erschienen 1984 bei Walter de Gruyter & Co., Berlin 30
ISBN 3-11010-325-7

<3> Goldschlager / Lister: Informatik Eine moderne Einführung
Übersetzt von Dieter Ballin
Erschienen 1984 im Carl Hanser Verlag München Wien
ISBN 3-44613-952-4