B 2226 F

LOG IN 3

Heft 3
3. Jahrg. 1983

Informatik in Schule und Ausbildung

Oldenbourg

Thema:
Programmiersprachen

LOGO

Michael J. Tauber

Einleitende Bemerkungen

LOGO ist eine interaktive, listenverarbeitende, funktionale Programmiersprache, die zusätzlich mit einer Turtlegraphik (turtle = Schildkröte) ausgestattet ist. Zur Gestaltung der Turtlegraphik sowie der Spracharchitektur von LOGO hat vor allem der Lernpsychologe Papert (Massachusets Institute of Technology) beigetragen. Grundideen für die Arbeit mit LOGO sind u.a.

- Unterstützung von entdeckendem, spielerischen Lernen
- Formulieren vonVermutungen, Ausprobieren derselben
- Problemlösen mit Hilfe von Prozeduren; Zerlegung von Problemen in Teilprobleme
- Erweiterung der Sprache durch eigene problemspezifische Prozeduren; Baukastenprinzip
- Erarbeitung von programmsprachlichen Konzepten und Entwicklung von Problemlösung über das Arbeiten mit der Turtlegraphik
- einfaches interaktives Arbeiten

Das Programmiersystem LOGO besteht aus einer listenverarbeitenden Sprache und zusätzlichen Systemteilen, wie *Editor, File-System, Turtlegraphik* und *Programmierhilfen.* Im folgenden wird das Apple-LOGO zugrunde gelegt.

Datenobjekte: Wörter, Zahlen, Listen

Programmieren mit LOGO bedeutet die Ausführung von Operationen auf Datenobjekten vom Typ WORT, ZAHL oder LISTE. Datenobjekte vom Typ Wort beginnen mit einem " gefolgt von einer Zeichenkette, bestehend aus keinem (leeres Wort), einem oder mehreren Zeichen. Beispiele sind:

"JULIA "SEBASTIAN "SUSI "KATHARINA
"KINDER ". . . .

Eine Zahl ist ein spezielles Wort, das nur aus Ziffern (und einem Vorzeichen) besteht und ganzzahlig oder als Dezi-

malzahl (Festkomma oder Gleitkomma) geschrieben werden kann. Beispiele für korrekte Zahlen sind:

1 34 25.4567 3.14 2.3E5 −67.3 −4 23.5N7 ...

Listen hingegen sind eine geordnete Zusammenfassung von einzelnen Datenobjekten zu einem neuen Datenobjekt. Geordnet bedeutet im Gegensatz zum mathematischen Mengenbegriff, daß die Reihenfolge der einzelnen Listenelemente von Bedeutung ist. Als Listenelemente können Wörter, Zahlen oder wiederum Listen fungieren. Auch die leere Liste ist definiert. Listen werden durch Aufzählung der Listenelemente in ihrer Reihenfolge, wobei Wörter ohne '' geschrieben werden, und durch Einfassung von [ ] gekennzeichnet. Beispiele für Listen sind:

[JULIA SEBASTIAN KATHARINA SUSI LUCIA EDITH]
... Liste von Kindern

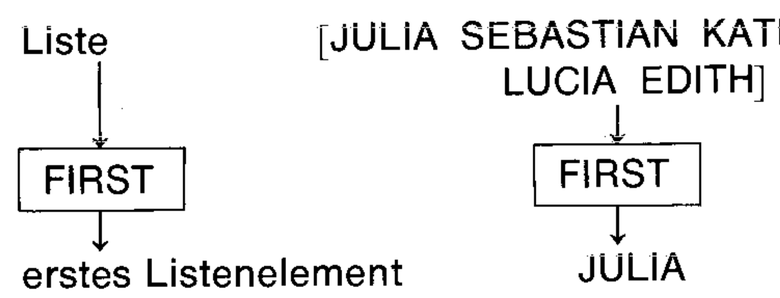
[[JULIA SEBASTIAN] [KATHARINA SUSI LUCIA] EDITH]
... Liste von Geschwistern; Kinder, die wieder in einer Liste zusammengefaßt sind, sind Geschwister

[4.5 1.2 3 4 6 7.3] ... Liste des Alters der Kinder aus dem ersten Beispiel

[ ] ... leere Liste

Prozedurenkonzept

Prozeduren führen Operationen auf den Datenobjekten Wort, Zahl oder Liste aus. LOGO ist eine funktionale Programmiersprache, d.h. jede Prozedur (Funktion) benötigt eine feste Anzahl von Datenobjekten zur Eingabe und produziert ein Datenobjekt als Ausgabe (im Sinne eines Resultates). Die Sprachelemente von LOGO sind solche Prozeduren (Standardprozeduren). Die Standardprozedur FIRST möge das funktionale Prozedurenkonzept verdeutlichen. FIRST verlangt als Eingabe eine Liste (Wort) und produziert als Ergebnis das erste Element der Liste (des Wortes). Zur Veranschaulichung des funktionalen Zusammenhangs ist eine „Box"-Darstellung recht hilfreich:

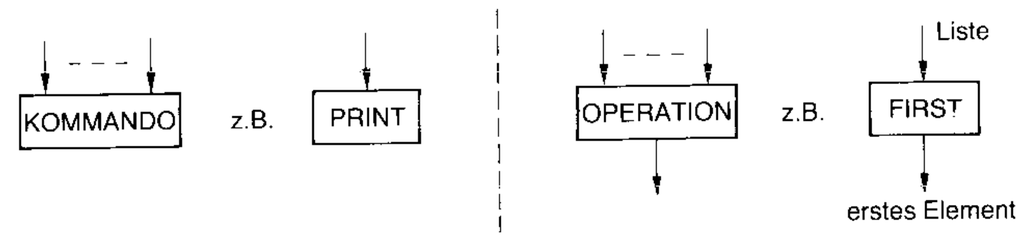


Weitere Beispiele für die Anwendung der Prozedur FIRST sind:

FIRST [[JULIA SEBASTIAN] [KATHARINA SUSI LUCIA] EDITH] produziert [JULIA SEBASTIAN]

FIRST [4.5 1.2 3 4 6 7.3] produziert 4.5

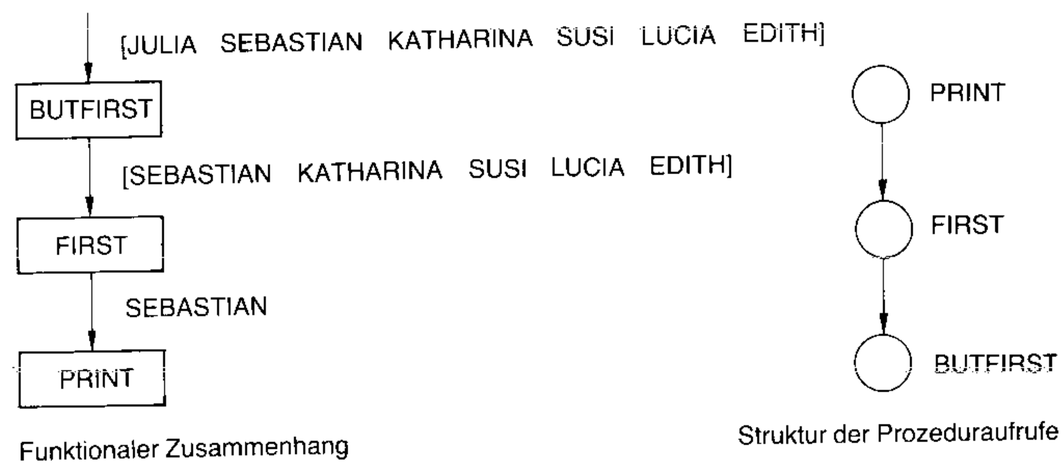
LOGO erwartet, daß das von einer Prozedur als Ausgabe (Resultat) produzierte Datenobjekt für eine neue Prozedur als Eingabe dient. Daher antwortet etwa Apple-LOGO auf FIRST [JULIA SEBASTIAN SUSI] mit „I dont't know what to do with JULIA". Auf PRINT FIRST [JULIA SEBASTIAN SUSI] antwortet LOGO jedoch mit JULIA. PRINT ist eine spezielle Prozedur, die nur eine Eingabe verlangt und keine Ausgabe im Sinne eines Resultates produziert. (PRINT führt zwar zu einer physikalischen Ausgabe, das eingegebene Datenobjekt erscheint am Bildschirm; im funktionalen Sinne produziert PRINT jedoch kein Datenobjekt als Resultat zur weiteren Verarbeitung.) Solche Prozeduren heißen in der LOGO-Diktion KOMMANDOS im Gegensatz zu OPERATIONEN, die auch eine Ausgabe produzieren. Folgende Skizze möge dies veranschaulichen:



Arbeiten mit LOGO bedeutet Zusammensetzen von Prozeduren (Zusammensetzung von Funktionen), wobei die äußerste Prozedur ein Kommando ist. Ein Beispiel für diese Vorgehensweise ist zum Beispiel die Klärung der Frage, welches Kind an der zweiten Stelle in der Reihe steht. Dazu benötigen wir eine weitere Standardprozedur BUTFIRST, die zu einer Liste (Eingabe) eine Liste ohne das erste Element der Eingabeliste produziert. Die Frage nach dem zweiten Kind lautet nun:

PRINT FIRST BUTFIRST [JULIA SEBASTIAN KATHARINA SUSI LUCIA EDITH]. LOGO antwortet mit SEBASTIAN.

Die Boxdarstellung soll den funktionalen Zusammenhang noch einmal verdeutlichen.



Weitere Standardprozeduren auf Listen

Im folgenden wird eine Auswahl weiterer listenverarbeitender Standardprozeduren von LOGO vorgestellt und ihre Wirkung anhand unserer Kinderlisten veranschaulicht.

- LAST beantwortet die Frage nach dem letzten Element in der Liste;
  LAST [JULIA SEBASTIAN SUSI] produziert SUSI
- BUTLAST produziert eine Liste ohne das letzte Element;
  BUTLAST [JULIA SEBASTIAN SUSI] produziert [JULIA SEBASTIAN]
- SENTENCE produziert eine Liste aus allen Wörtern, die als Eingabeobjekte einzeln oder in Listen vorkommen;
  SENTENCE [KATHARINA EDITH] [JULIA SEBASTIAN SUSI] produziert [KATHARINA EDITH JULIA SEBASTIAN SUSI]; SENTENCE "LUCIA [[JULIA] EDITH] produziert [LUCIA JULIA EDITH]
- LIST hingegen bildet eine Liste zusammengesetzt aus den Eingabeobjekten; LIST [JULIA] [SUSI EDITH] produziert [[JULIA] [SUSI EDITH]]
- COUNT produziert die Anzahl von Objekten in einer Liste; COUNT [JULIA SEBASTIAN SUSI LUCIA EDITH] produziert 5.

Arithmetische Grundoperationen

LOGO gestattet die Formulierung von arithmetischen Operationen in Prefix-Notation (z.B. SUM 2 3 produziert 5)

sowie in Infix-Notation (z. B. 2 + 3, 4 * 2.8, 3/7, 7–4). Ferner stellt LOGO die wichtigsten mathematischen Funktionen (SIN, COS, SQRT, . . .) zur Verfügung.

Schreiben eigener Prozeduren, Erweiterung von LOGO

Selbstgeschriebene Prozeduren haben in LOGO dieselbe logische Bedeutung wie Standardprozeduren. Damit kann sich der Benutzer für jede Anwendung eine eigene Sprache konstruieren (Baukastenprinzip).

Für jede Prozedur sind bei ihrer Definition die Zahl der Eingabeobjekte, sowie das Ausgabeobjekt, das die Prozedur produziert, festzulegen. Bei Kommandos entfällt das Ausgabeobjekt. Anhand des oben gestellten Problems, das zweite Kind in einer Reihe zu bestimmen, wird die Definition einer neuen Prozedur exemplarisch erläutert:

```
TO SECOND :OBJEKT
OUTPUT FIRST BUTFIRST :OBJEKT
END
```

Die Definition einer neuen Prozedur beginnt mit TO, gefolgt vom Namen der Prozedur und zumeist einer Aufzälung der Eingabeobjekte in Form von *Platzhaltern*, die als Eingabevariable zu interpretieren sind. Die Platzhalter beginnen mit : gefolgt von einem Namen für die Platzhalter. Beispiele sind:

:KINDER :OBJEKT :A :X . . .

In derselben Notation werden Platzhalter innerhalb der Prozedur verwendet. Die Namen der Platzhalter haben nur eine *lokale* Bedeutung. Variablen gleichen Namens in verschiedenen Prozeduren leben in verschiedenen Welten. Bei Aufruf der Prozedur werden die Platzhalter durch konkrete Datenobjekte ersetzt. So gilt dann für die eben definierte Prozedur SECOND

SECOND [JULIA LUCIA SEBASTIAN] produziert LUCIA

Das Kommando OUTPUT in unserer Prozedur SECOND macht das dahinter geschriebene Datenobjekt (in unserem Beispiel FIRST BUTFIRST :OBJEKT) zum Ausgabeobjekt der Prozedur und beendet gleichzeitig die Prozedur, d. h. OUTPUT gibt die Kontrolle an die aufrufende Prozedur zurück. Bei Kommandos bewerkstelligt die Prozedur STOP das Beenden der Prozedur sowie die Rückgabe der Kontrolle an die aufrufende Prozedur. Mit END ist die Definition der Prozedur abgeschlossen. Die Prozedur SECOND kann nun genauso wie Standardprozeduren verwendet werden. Beispiele sind:

SECOND BUTFIRST [JULIA SUSI LUCIA]
produziert LUCIA
SECOND BUTFIRST BUTFIRST [JULIA SUSI]
produziert Error
PRINT SECOND [KATHARINA SUSI] druckt SUSI am Bildschirm

Rekursion, Kontrollstrukturen

Zur Illustration weiterer Eigenschaften von LOGO wollen wir folgendes Problem lösen:

Kinder sind in einer Reihe (Kreis) aufgestellt. Sie zählen mit einem Abzählreim durch. Ist der Reim noch nicht durchgezählt und das letzte Kind in der Reihe erreicht, beginnt das Weiterzählen wieder am Anfang der Reihe. Das mit der letzten Silbe des Reimes angezählte Kind scheidet aus der

Turtlegraphik

Programmieren und Problemlösen; Lernen und Entdecken durch Generierung geometrischer Objekte

Die folgenden Beispiele wurden von 12–15jährigen Schülern im Rahmen des ersten österreichischen Computercamps (Ferien und Computer) entwickelt und spiegeln die wachsenden Erfahrungen und Erkenntnisse der Schüler wider.

Es werden die Kommandos FORWARD (abgekürzt FD), RIGHT (abgekürzt RT), LEFT (abgekürzt LT) und SETHEADING 0 (dreht die Turtel nach „Norden") benutzt.

```
FD 30 RT 90
FD 30  LT 90
FD 30 RT 90
FD 30 LT 90
```

```
TO TREPPE
REPEAT 2 [FD 30 RT 90 FD 30 LT 90]
END
```

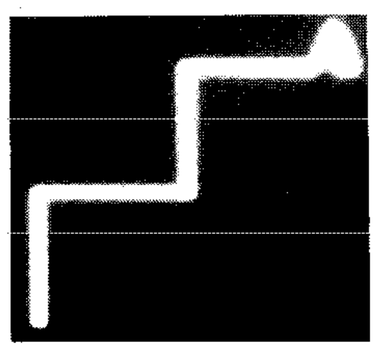
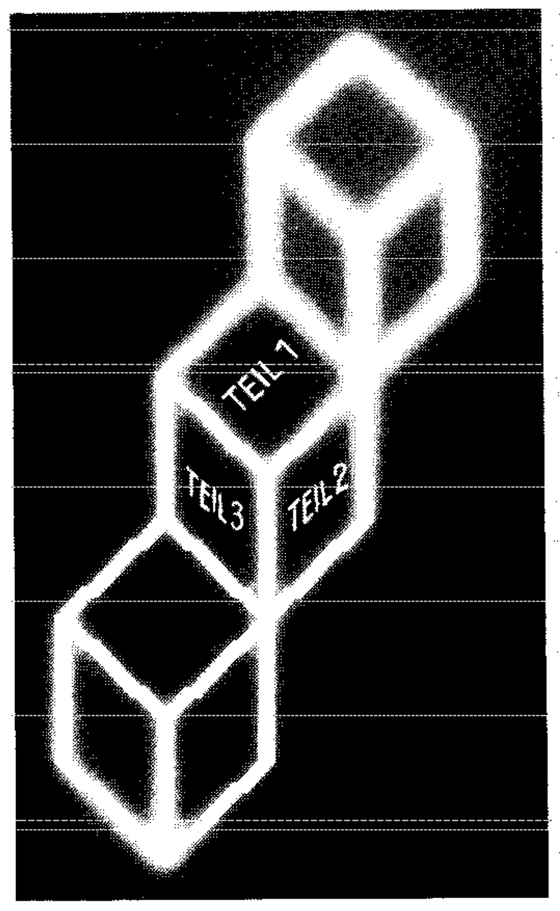


Bild 1: Nach anfänglichem „wahllosem Zeichnen" entstehen die ersten systematisch beschreibbaren Objekte, wie etwa diese Treppe.




```
TO TEIL 1
RT 45
REPEAT 4 [FD 30  LT 90]
END

TO TEIL 2          TO TEIL 3
FD 30 RT 45        FD 30 LT 45
FD 30 RT 135       FD 30 LT 135
FD 30 RT 45        FD 30 LT 45
FD 30              FD 30
END                END

TO WUERFEL
TEIL 3
SETHEADING 0 TEIL 2
SETHEADING 0 FD 30
TEIL 1
END

TO TREPPEN
REPEAT 3 [WUERFEL FD 30 SETHEADING 0]
END
```

Bild 2: Diese „perspektivische" Treppe zeigt, daß bei zunehmender Komplexität des Problems dieses durch die Definition von Teilproblemen leichter zu lösen ist. TEIL 1, TEIL 2, TEIL 3 beschreiben je 1 Seite; WUERFEL faßt diese Seiten zusammen und TREPPEN wiederholt diesen Vorgang.

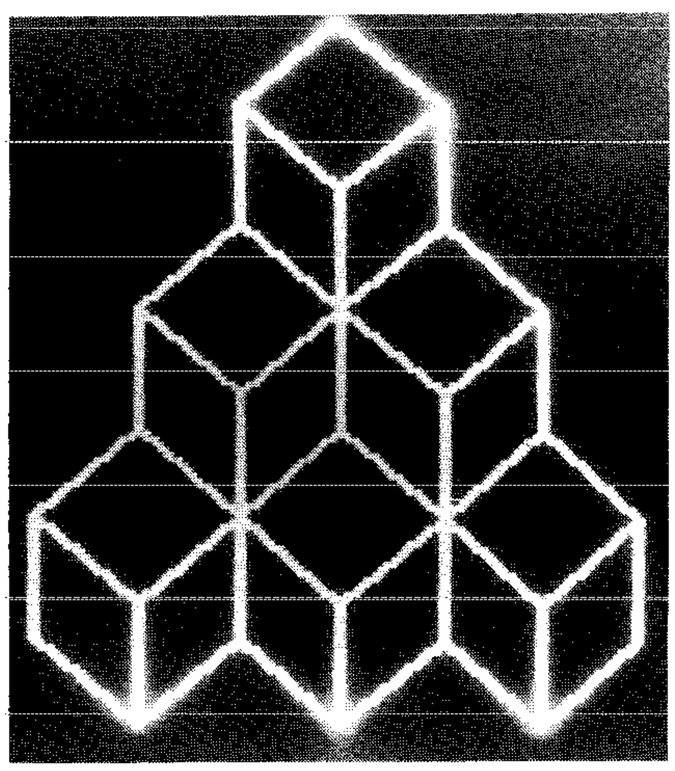
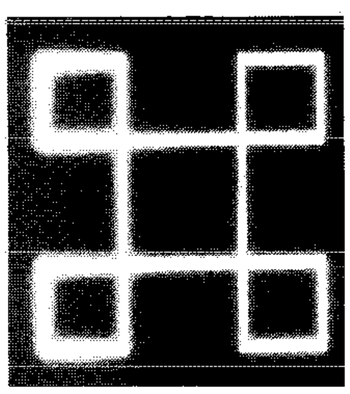
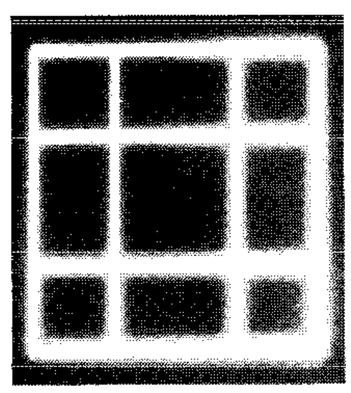


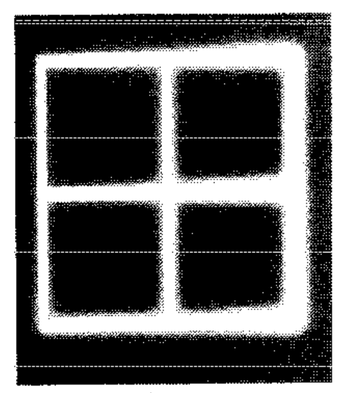
Bild 3: Die Treppe war übrigens auch nur ein Baustein. Ziel war die Generierung einer komplexen Treppenstruktur (ein zwölfjähriges Mädchen hatte die Idee und programmierte dieses Bild).



FIGUR 15 45



FIGUR 45 15

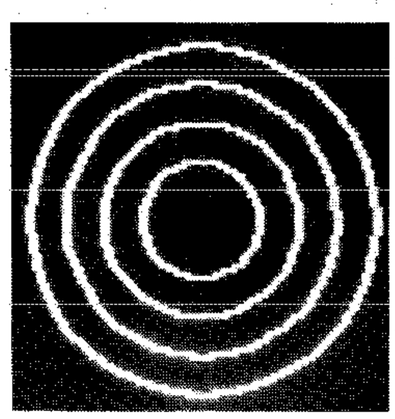


FIGUR 30 30

```
TO FIGUR :A :B
REPEAT 4 [BLOCK :A :B]
END
```

```
TO BLOCK :A :B
REPEAT 3 [FD :A LT 90]
FD :B
END
```

Bild 4: Der Wunsch, Figuren in verschiedenen Größen zu generieren, führt zum Begriff der Variablen. Dabei entdecken die Schüler, daß bei verschiedener Wahl der Eingabeobjekte unterschiedliche Figuren entstehen. Ausgangsfrage war lediglich die Generierung der linken Figur.



```
TO RINGE :RADIUS :SCHRITT
CIRCLE :RADIUS
PENUP
LEFT 90
FORWARD :SCHRITT
RIGHT 90
PENDOWN
RINGE :RADIUS +
:SCHRITT :SCHRITT
END
```

Bild 5: In einem Programm unterschiedlich große Figuren gleicher Bauart zeichnen zu wollen, führt zur Rekursion. Dies ist am Beispiel dieser „Kreisringe" zu sehen.

Reihe aus. In der neuen Reihe (Kreis) wird wieder durchgezählt. Das Verfahren wird solange wiederholt, bis nur mehr ein Kind überbleibt.

Die Ermittlung dieses Kindes soll eine Prozedur SIEGER leisten, deren Eingabeobjekte die Kinderreihe, sowie der Abzählreim sind. Als Ausgabe soll SIEGER das übriggebliebene Kind produzieren. Innerhalb der Prozedur SIEGER müssen die um ein Kind jeweils reduzierten Reihen solange durchgezählt werden, bis nur mehr ein Kind zurückbleibt. Als grundlegende Kontrollstruktur verwendet LOGO die REKURSION, so daß im ersten Ansatz die Prozedur so aussieht:

```
TO SIEGER :REIM :LISTE
OUTPUT SIEGER :REIM DURCHZAEHLEN :REIM :LISTE
END
```

produziert Kinderreihe
ohne ausgeschiedenem Kind

Die noch zu definierende Prozedur DURCHZAEHLEN :REIM :LISTE zählt mit einem Abzählreim eine Liste durch und produziert als Ausgabe eine Liste, aus der das zuletzt angezählte Kind ausgetreten ist.

Das wiederholte Durchzählen der jeweils um ein Kind reduzierten Reihe wird durch rekursive Aufrufe der Prozedur SIEGER bewerkstelligt, doch muß noch sichergestellt sein, daß die Aufrufe dann abbrechen, wenn die Prozedur DURCHZAEHLEN nur mehr eine Reihe bestehend aus einem Kind produziert.

Zur Formulierung von Kontrollstrukturen stellt LOGO spezielle Prozeduren (PRÄDIKATE) zur Verfügung, die als Ausgabe die Objekte TRUE oder FALSE produzieren. Standardprozeduren mit einem nachgestellten P sind als Prädikate erkennbar. So verlangt etwa EQUALP zwei Datenobjekte (Wörter oder Zahlen oder Listen) als Eingabe und produziert TRUE, falls die beiden gleich sind, andernfalls FALSE. Die Abbruchbedingung für die rekursiven Aufrufe kann nun mit der Standardprozedur IF bewerkstelligt werden, die formal folgendes Aussehen hat:

IF Prädikt Liste 1 Liste 2

Liste ist eine Liste von Prozeduren, die ausgeführt wird, wenn das Prädikat TRUE produziert; Liste 2 wird ausgeführt, wenn das Prädikat FALSE produziert.
Liste 2 kann entfallen.

Wir sind nun in der Lage unsere Prozedur SIEGER fertig zu definieren:

```
TO SIEGER :REIM :LISTE
IF EQUALP COUNT :LISTE » [OUTPUT FIRST :LISTE←
OUTPUT SIEGER :REIM DURCHZAEHLEN :REIM :LISTE
END
```

Das einmalige Durchzählen einer Liste mit einem Abzählreim geschieht nach folgender Modellvorstellung. Das Kind, das gerade eine Silbe des Reimes ansagt, steht vor dem Ansagen an erster Stelle der Liste und tritt dann an das Ende der Liste. Diesen Prozeß beschreibt die Prozedur ERSTLETZT:

```
TO ERSTLETZT :LISTE
OUTPUT SENTENCE BUTFIRST :LISTE FIRST :LISTE
END
```

Das Wechseln vom Anfang an das Ende der Liste (kurzes Hinübertreten in einem „offenen" Kreis) nach Aufsagen einer Silbe wird solange durchgeführt, bis der gesamte Abzählreim durchgezählt ist. Die Liste :REIM wird dabei bei jedem rekursiven Aufruf der Prozedur ERSTLETZT (Wechsel vom Anfang an Ende) um die jeweils erste Silbe vermindert, bis die Liste leer ist. Diese Ideen führen zur Definition der Prozedur DURCHZAEHLEN :REIM :LISTE.

```
TO DURCHZAEHLEN :REIM :LISTE
IF EMPTYP :REIM [OUTPUT BUTLAST :LISTE]
OUTPUT DURCHZAEHLEN BUTFIRST :REIM
            ERSTLETZT :LISTE
END
```

Zu Zeile 2: An der *letzten* Stelle steht das ausgezählte Kind, das ausscheidet.

Mit der Prozedur SIEGER :REIM :LISTE kann man nun für beliebige Auszählreime und beliebige Kinderreihen den Sieger durch „Auszählen" ermitteln. So kommt es etwa zu folgenden Ergebnissen:

SIEGER [AM DAM DES] [JULIA SEBASTIAN KATHARINA SUSI LUCIA] produziert SUSI

SIEGER [AM DAM DES DIE SE MA LE BES DIE SE MA LE PUM PER NES AM DAM DES] [JULIA SEBASTIAN KATHARINA SUSI LUCIA EDITH] produziert JULIA

Ein neues Problem ist etwa, vor dem Auszählen die Kinder in einer bestimmten Reihenfolge aufzustellen. Eine Möglichkeit ist die Reihenfolge in der Reihe genau umzudrehen, womit man ein sehr gutes Beispiel für die Anwendung des Rekursionsprinzips gewonnen hat. Die Grundidee für eine solche Prozedur UMKEHR ist die, daß das letzte Kind in der Reihe an die erste Stelle einer neuen Reihe gestellt werden und der Rest der neuen Reihe durch Anwendung desselben Prinzips auf die alte Reihe ohne das letzte Element rekursiv ermittelt wird. Dies geschieht solange, bis die alte Reihe aufgebraucht ist. Die Prozedur UMKEHR :LISTE lautet daher:

```
TO UMKEHR :LISTE
IF EMPTYP :LISTE [OUTPUT :LISTE]
OUTPUT SENTENCELAST :LISTE UMKEHR BUTLAST
                                  :LISTE
END
```

Variable, Wertzuweisung

Das im Zusammenhang mit der Einführung des Prozedurenkonzeptes erläuterte Prinzip der Platzhalter ist ein Beispiel für die Verwendung von Variablen in LOGO. Dieser Typ von Variable wird bei Aufruf der Prozedur mit einem konkreten Wert belegt. Die Platzhalter sind Variable, die nur lokale Bedeutung haben. Bei Aufruf wird der Platzhalter mit einem konkreten Wert belegt, nach Beendigung der Prozedur steht der Wert des Platzhalters nicht mehr zur Verfügung.

Will man jedoch eine Variable mit einem Wert belegen, der für mehrere Prozeduren oder auch nach Beendigung einer Prozedur noch für längere Zeit zur Verfügung stehen soll, oder einem Datenobjekt einen Namen zuweisen, so erreicht man dies mit der Prozedur MAKE. Die Wirkung dieser Prozedur wird durch einige Beispiele in verschiedenen Sprachweisen erläutert.

MAKE "X 35 weist dem Objekt "X den Wert 35 zu
MAKE "KINDER [JULIA SEBASTIAN SUSI EDITH] gibt der Kinderliste den Namen KINDER;
MAKE "REIHE "KINDER weist dem Objekt KINDER den Namen REIHE zu

Auf den Wert einer Variablen greift man über die Prozedur THING zu:

THING "KINDER produziert [JULIA SEBASTIAN SUSI EDITH]; THING THING "REIHE hat dasselbe Ergebnis;
PRINT SUM THING "X 5 zeigt am Bildschirm die Zahl 40

Anstelle von THING kann auch wie bei der Definition von Platzhaltern eine dem Variablennamen vorangestellter : verwendet werden, um auf den Wert der Variablen zuzugreifen

PRINT :KINDER druckt JULIA SEBASTIAN SUSI EDITH
SUM :X :X+3 produziert 73

Man beachte die klare Unterscheidung zwischen Namen und Wert eines Datenobjektes; im BASIC etwa durch Verwendung des = –Zeichens eher mißverständlich (I=I+ 1 im BASIC versus MAKE "I :I+1 in LOGO).

Prozeduren als Daten

Listenverarbeitende Sprachen kennen keine logische Trennung von Programmen und Daten. So bewirkt die Prozedur RUN, die eine Liste bestehend aus Prozeduren als Eingabeobjekt verlangt, daß diese Liste bei Ausführung der Prozedur ausgeführt wird. Dies erlaubt zum Beispiel, daß Platzhalter von Prozeduren bei Aufruf auch durch LOGO-Prozeduren und nicht nur durch Datenobjekte belegt werden können.

Das Beispiel der Erstellung einer Wertetabelle für verschiedenste Funktionen soll dies verdeutlichen.

```
TO TABELLE :FUNKTION :X :XEND
IF :X = :XEND [STOP] [PRINT :X RUN :FUNKTION)]
TABELLE :FUNKTION :X+1 :XEND
END
```

TABELLE [:X+2] 1 5 druckt z.B.:
1 3
2 4
3 5
4 6

Weitere mögliche Prozeduraufrufe sind
TABELLE [SIN :X] 1 10 oder
TABELLE [SQRT SIN :X + COS :X] 1 20.

Turtlegraphik

Mit Hilfe der Turtlegraphik kann der LOGO-Benutzer einerseits Graphik erstellen und andererseits auch Grundprinzipien des Programmierens erlernen. Kernstück der Turtlegraphik ist die *Turtle (Schildkröte)*, die durch ihre programmierbaren Bewegungen Graphik am Bildschirm erzeugt.

Die TURTLE ist ein Pfeil, der auf einer aktuellen Postion steht und mit einer aktuellen Richtung versehen ist. Position und Richtung können relativ zur momentanen Lage verändert werden. Ausgangslage (HOME) ist die Mitte des Bildschirmes.

Die Prozedur FORWARD bewegt die Turtle um eine anzugebende Wegstrecke in Pfeilrichtung, die Prozedur BACKWARD entgegensetzt zur Richtung, die Prozedur RIGHT dreht die Richtung der Turtle um einen anzugebenden Winkel nach rechts, die Prozedur LEFT macht dies analog nach links. FD, BK, RT, LT sind Abkürzungen für die eben eingeführten Prozeduren.

Am Beginn des Operierens mit der Turtlegraphik steht die Turtle in der Mitte des Bildschirms und zeichnet gemäß ihren durch die Prozeduren FD, BK, RT oder LT gesteuerten Bewegungen eine „Spur" auf den Schirm. Der „Zeichenstift" kann durch PENUP „angehoben" und damit außer Kraft gesetzt werden, so daß einzelne Bewegungen der Turtle keine „Spuren" hinterlassen und beginnt nach Aufruf der Prozedur PENDOWN wieder zu zeichnen. Mit HOME kehrt die Turtle wieder zu ihrem Ausgangspunkt in die Mitte des Bildschirmes zurück. Ferner kann die Turtle auf ihrem Weg mit der Funktion eines „Radiergummis" versehen werden. Daneben können Position und Richtung der Turtle auch absolut bestimmt werden (Zur Turtlegraphik siehe die Beispiele in dem separaten Teil Turtle-Graphik).

Literatur

[1] Abelson, H.: LOGO for the APPLE II. Hightstown, NJ; BYTE/McGraw Hill, 1982.
[2] Abelson, H.; diSessa, A.: Turtle Geometry. Cambridge, MIT-Press 1981.
[3] Abelson, H.: Einführung in LOGO. IWT, Vaterstetten 1983.
[4] BYTE 7, Nr. 8, August 1982; Sonderheft LOGO.
[5] Fischer, G.; Böcker, H.D.: Arbeitsmaterialen zum interaktiven Programmieren. HIBS, Wiesbaden.
[6] Goldenberg, P.: Special Technology for Special Children, Baltimore 1979.
[7] Kidall, G.; Thonburg, D.: Digital Research's DR LOGO, BYTE 8, Juni 1983.
[8] Löthe, H.: Das LOGO-System und seine Implementationen auf dem TI99/4 und APPLE II. LOG IN 2(1982) Heft 3.
[9] Löthe, H. et al.: Diverse LOGO-Berichte. Päd. Hochschule Esslingen.
[10] Papert, S.: Mindstorms – Kinder, Computer. Neues Lernen. Birkhäuser, Basel 1982.
[11] Ross, P.: LOGO Programming. Addison-Wesley, London 1983.
[12] Tauber, M.J.. Schauer, H.: Beispiele aus dem Computercamp. OCG Wien 1983.
[13] Thonburg, D.: Discoveries of Beauty: An Invitation to LOGO, Addison-Wesley, Reading Mass. 1983.
[14] Watts, d.: Learning with LOGO. BYTE McGrawHill, Petersborough 1983.

Dr. Michael J. Tauber
z.Z. IBM Heidelberg
Tiergartenstr. 15
6900 Heidelberg

Implementationen

1. APPLE LOGO; Vertrieb: APPLE Computer Inc.; Hersteller: Logo Computer Systems Inc. (LCSI); Computer: APPLE II
2. TERRAPIN LOGO; Hersteller: Terrapin Inc., Cambridge, Mass.
   KRELL LOGO; Hersteller: Krell Software Corp., Stony Brook N.J. Beinahe identische Versionen des MIT-LOGO, ähnlich dem APPLE LOGO, Computer: APPLE II
3. IWT-LOGO; Hersteller und Vertrieb: IWT-Verlag, Dahlienstr. 4, 8011 Vaterstetten; deutsche Übersetzung des MIT-LOGO; entspricht den Versionen Terrapin und Krell; Computer: APPLE II, BASIS 108
4. TI-LOGO; Hersteller und Vertrieb: Texas Instruments Inc. Keine Gleitkommarechnung, keine fortgeschrittene Listenverarbeitung, dafür Ausnutzung von Animationshardware, Computer: TI99/4 Anwendung eher für Primarstufe
5. TRS-80 COLOR LOGO; Hersteller: Radio Shack Corp. Anwendung eher für die Primärstufe
6. DR LOGO; Hersteller: Digital Research, POB 579 Pacific Grow, CA 93950, komfortable Erweiterung von APPLE LOGO, gemischte Text/Graphik Anwendung, einfache und doppelte Genauigkeit, Textfenster für Editieren, Fehlermeldungen und ProgrammTrace, ..., alle wichtigen transzendenten Funktionen, Computer: Mikrocomputer mit CPIM, z. B. IBM PC.
7. EDINBURGH LOGO, Vertrieb: Research Machines Ltd. Microcomputer Systems, P.O. Box 75, Oxford, OXZ OBW, U.K.. LOGO Version mit etwas veränderter Syntax gegenüber dem MIT-LOGO, entwickelt von der AI-Group in Edinburgh, z.B. mit erweiterten Kontrollstrukturen (DO...UNTIL, WHILE ...); Computer: 380Z von Research Machines.
8. Portables LOGO von BBN (LOGO-S). Implementationen u.a. auf Rechnern von DEC. Anfrage nach implementierten Versionen oder nach Zusendung von Materialen zur Generierung einer portablen Version: Director Wallace Feurzeig, Educational Technology Dept., Bolt, Beranek and Newman Inc., 50 Moulton Street, Cambridge, MA 02238/USA
9. ATARI LOGO; Hersteller: Logo Computer Systems, Inc. (LCSI); Vertrieb: ATARI, Hamburg; Syntax wie APPLE LOGO, z. B. 4 Turtles, die in verschiedenen Gestalten definiert werden können, sowie 256 Farben; Computer ATARI 6pp XL
10. Weitere LOGO-Versionen sind angekündigt, z.B. für Commodore 64

Ausgewählte Projektgruppen

1. LOGO GROUP, Artificial Intelligence Lab., MIT Boston, 545 Technology Square, Cambridge, MA 02139
2. PROKOP (abgeschlossen), Hessisches Institut für Bildungsplanung und Schulentwicklung, Bodenstedtstr. 7, 6200 Wiesbaden
3. LOGO-Projekt der Pädagogischen Hochschule Esslingen, Prof. Löthe, Flandernstr. 101, 7300 Esslingen
4. LOGO im Rahmen der österreichischen Computercamps, Gesellschaft für Bildungstechnologie, Dr. M.J. Tauber, A-4173 St. Veit 32 sowie Österreichische Computergesellschaft, Dr. H. Schauer, TU Wien, A-1040 Wien
5. Artificial Intelligence Department, University of Edinburgh, Edinburgh, U.K.
6. British LOGO User Group, c/o Mrs. P. Valley, 26 Tithby Road, Bingham, Nottingham, U.K. (mit USER Zeitschrift)
7. Weitere Namen von Projektgruppen über Director W. Feurzeig, Educational Dept., BBN, 50 Moulton Street, Cambridge, MA 02238/USA

LOGO-User Zeitschriften

TURTLE NEWS; LOGO NEWSLETTER, beide verlegt von YPLA (Young People's Logo Association, Inc.), 1208 Hillsdale Drive, Richardson, TX 75081, USA