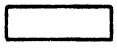
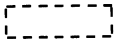
Anhang A

ELAN - systematischer Überblick

Hier wird versucht, eine kurze Übersicht über die Elemente der Programmiersprache ELAN zu geben.*) Nähere Einzelheiten und tieferliegende Probleme der Syntax und Semantik können an dieser Stelle nicht behandelt werden. Solchen Fragen kann man mit Hilfe der "ELAN - Sprachbeschreibung" und des "ELAN - Handbuchs" (s. Literaturhinweise) nachgehen.

Methode der Beschreibung:

- - hinterlegte Elemente erscheinen direkt im ELAN-Programm.
-  drückt Zusammenfassung zu einem Ganzen aus.
-  solche Teile dürfen auftreten, sind also optional.
- / bezeichnet zwei Alternativen. Eine davon muß gewählt werden.
- ... ... bezeichnet eine Folge (ohne Trennzeichen) von ein- oder mehrfachem Auftreten des vorangehenden Elements. Beispiel:

  Ziffer ... ...

  bedeutet eine Ziffernfolge. (Eine Ziffer allein ist auch eine Folge!)
- ,...,... bezeichnet eine Folge, deren Elemente durch Komma getrennt werden.
- ;...;... bezeichnet eine Folge, deren Elemente durch Semikolon getrennt werden.

Bei den einzelnen Begriffen erscheinen in () Verweise auf nähere Erläuterungen in diesem Buch. Außerdem werden die englischen Begriffe der ELAN-Sprachbeschreibung zusätzlich aufgeführt, falls sie sich nicht durch wörtliche Übersetzung ergeben.

---

*) Die verhältnismäßig selten gebrauchten "Prozeduren als Parameter" sind in dieser Beschreibung aus Gründen der Übersichtlichkeit nicht enthalten. Sie werden im Detail auf Seite 97 dieses Buches beschrieben.

1. Basiselemente

Namen (12) werden zur Bezeichnung von Datenobjekten, Refinements, Prozeduren, und Paketen verwendet. Sie dürfen Blanks enthalten.

Kleinbuchstabe [Kleinbuchstabe / Ziffer] ... ...

NAMEn (12) ("bold") dienen zur Bezeichnung von Datentypen und Operatoren. Sie dürfen keine Blanks enthalten und müssen sich von den reservierten Schlüsselworten der Sprache unterscheiden:

Großbuchstabe ... ...

OPNAMEn sind Bezeichner von Operatoren:

+ / - / * / / / ** / := / < / > / <= / >= / = / <> / $ / % / & / | / NAME

Denoter (18,20,26,43) dienen zur Denotation von Werten der elementaren Datentypen.

INT-Denoter:
REAL-Denoter: [e [+/-] Ziffer ... ...]
TEXT-Denoter: " "
BOOL-Denoter:

2. Ausdrücke

Ausdrücke (26) ("primary") können aus folgenden Grundelementen gebildet werden:

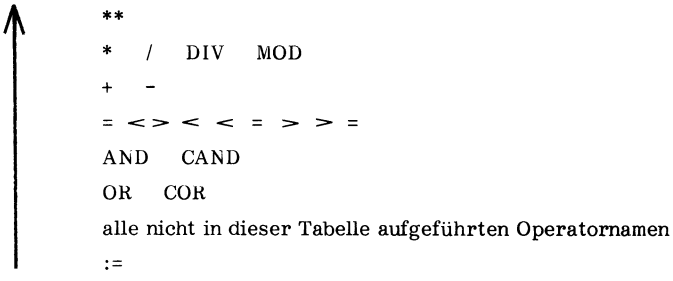
Denoter (s.o.)	
Datenobjekt	Name
Refinementaufruf	Name
Prozeduraufruf	Name [(Ausdruck ,... ,...)]
Subskription	ROW-Ausdruck [INT-Ausdruck]
Selektion	STRUCT-Ausdruck . Name
Konstruktor	Typ : (Ausdruck ,... ,...)
Konkretisierer	CONCR (Ausdruck)

Diese Grundelemente können nach bestimmten Regeln Ausdrücke bilden:

Ein Grundelement selbst ist schon ein Ausdruck.

dyadische Formel	Ausdruck OPNAME Ausdruck
monadische Formel	OPNAME Ausdruck
Klammerung	(Ausdruck)

Monadische Operatoren werden immer zuerst ausgewertet. Die Priorität der dyadischen Operatoren ergibt sich aus dem Namen:



Ausdrücke können

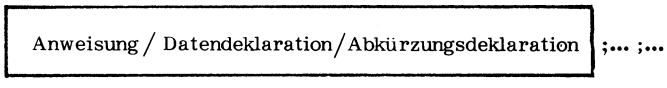
a) Werte liefern ('3 + 4') ,

b) Datenobjekte liefern ('feld (i)'),

c) nichts liefern ('x := 3' oder 'i INCR 2'). Auch diese Operationen werden aus Gründen der Einfachheit und Systematik als Ausdrücke bezeichnet.

3. Abschnitt

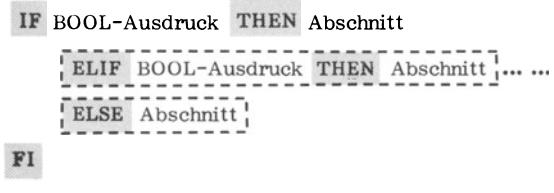
Ein Abschnitt (10) ("section") ist in erster Linie eine Folge von sequentiell auszuführenden Anweisungen. Er darf auch Deklarationen von Datenobjekten und Abkürzungen enthalten:



4. Anweisungen

Die einfachste Anweisung ("unit") ist ein Ausdruck. Damit sind u.a. auch Zuweisungen schon erfaßt. Weitere Anweisungen:

IF-Auswahl (16,50,109) :



Die SELECT-Auswahl wurde in diesem Buch nicht behandelt. Sie ermöglicht eine einfache Auswahl aus einer Serie mit Nummern bezeichneter Alternativen. Anhand des SELECT-Ausdrucks wird der entsprechende Fall (CASE) angewählt, falls er vorhanden ist - andernfalls der OTHERWISE-Teil:

```
SELECT INT-Ausdruck OF
    CASE Nummer ,...,... : Abschnitt ... ... *)
    OTHERWISE Abschnitt
ENDSELECT
```

Die Wiederholung (12,24,56) umfaßt die verschiedenen Formen der Schleife in allen Kombinationen:

```
FOR INT-Variable FROM INT-Ausdruck UPTO/DOWNTO INT-Ausdruck
WHILE BOOL-Ausdruck
REP
    Abschnitt
UNTIL BOOL-Ausdruck
ENDREP
```

Die LEAVE-Anweisung (52,92) ("terminator") dient zum irregulären Verlassen eines Refinements, einer Prozedur oder eines Operators:

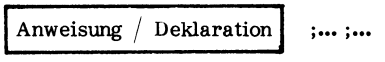
```
LEAVE Name / OPNAME WITH Ausdruck
```

*) 'Nummer' ist entweder ein INT-Denoter oder ein als "Abkürzung" deklarierter Name, der einen INT-Denoter bezeichnet.

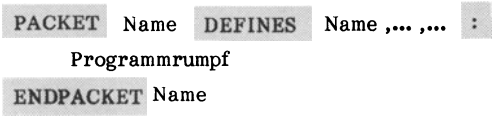
5. Programm und Paket

Ein Programm kann im einfachsten Fall aus nur einer Anweisung bestehen, im kompliziertesten Fall aus mehreren Paketen. (Im folgenden meint 'Deklaration' alle unter (7.) als '...-Deklaration' aufgeführten Alternativen.)

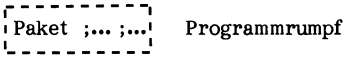
Programmrumpf ("main packet", "packet body"):



Paket (147):



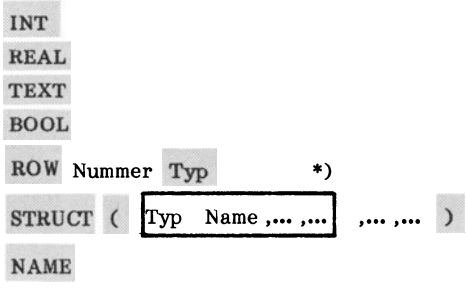
Programm:



6. Datentypen

Es gibt die elementaren Typen (18,20,26,43), die Reihungs- (55) und Verbundtypen (124) und schließlich noch abstrakte (neudefinierte) Typen (164).

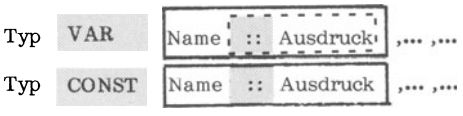
Typ:



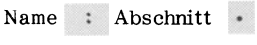
*) 'Nummer' ist entweder ein INT-Denoter oder ein als "Abkürzung" deklarierter Name, der einen INT-Denoter bezeichnet.

7. Deklarationen

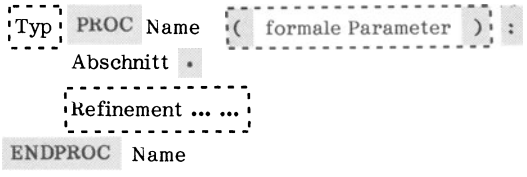
Datendeklaration (28,76,140) (Die Initialisierung ist bei Variablen optional)·



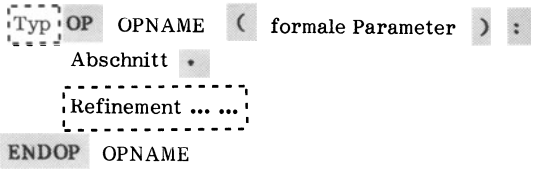
Refinementdeklaration (12,39):



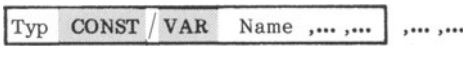
Prozedurdeklaration (76,81,89): (Falls die Prozedur einen Wert liefern soll, muß der Abschnitt des Rumpfes ein Objekt entsprechenden Typs liefern.)



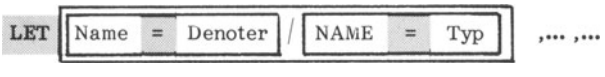
Operatordeklaration (163): (Operatoren haben immer ein oder zwei Parameter)



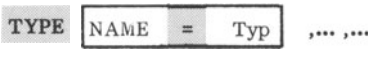
formale Parameter:



Abkürzungs-Deklaration (58,125) ("shorthand declaration"):



Typdeklaration:



Folgende Datentypen, Operatoren und Prozeduren stehen in einem ELAN-System zur Verfügung:

Datentyp INT

```
:= ..... (INT VAR ziel, INT CONST quelle)
maxint ..... --> INT
+ ..... (INT CONST links, rechts) --> INT
- ..... (INT CONST links, rechts) --> INT
* ..... (INT CONST links, rechts) --> INT
DIV ..... (INT CONST links, rechts) --> INT
MOD ..... (INT CONST links, rechts) --> INT
- ..... (INT CONST operand) --> INT
INCR ..... (INT VAR ziel, INT CONST increment)
DECR ..... (INT VAR ziel, INT CONST decrement)
= ..... (INT CONST links, rechts) --> BOOL
<> ..... (INT CONST links, rechts) --> BOOL
< ..... (INT CONST links, rechts) --> BOOL
<= ..... (INT CONST links, rechts) --> BOOL
> ..... (INT CONST links, rechts) --> BOOL
>= ..... (INT CONST links, rechts) --> BOOL
sign ..... (INT CONST operand) --> INT
SIGN ..... (INT CONST operand) --> INT
abs ..... (INT CONST operand) --> INT
ABS ..... (INT CONST operand) --> INT
** ..... (INT CONST basis, exponent) --> INT
min ..... (INT CONST a, b) --> INT
max ..... (INT CONST q, b) --> INT
random ..... (INT CONST untergrenze, obergrenze) --> INT
initializerandom (INT CONST startwert)
```

Datentyp TEXT

```
:= ..... (TEXT VAR ziel, TEXT CONST quelle)
= ..... (TEXT CONST links, rechts) --> BOOL
<> ..... (TEXT CONST links, rechts) --> BOOL
< ..... (TEXT CONST links, rechts) --> BOOL
<= ..... (TEXT CONST links, rechts) --> BOOL
> ..... (TEXT CONST links, rechts) --> BOOL
>= ..... (TEXT CONST links, rechts) --> BOOL
SUB ..... (TEXT CONST text, INT CONST pos) --> TEXT
subtext ..... (TEXT CONST quelle, INT CONST von, bis) --> TEXT
subtext ..... (TEXT CONST quelle, INT CONST von) --> TEXT
code ..... (TEXT CONST text) --> INT
code ..... (INT CONST code) --> TEXT
replace ..... (TEXT VAR ziel, INT CONST pos, TEXT CONST quelle)
text ..... (TEXT CONST quelle, INT CONST laenge) --> TEXT
text ..... (TEXT CONST quelle, INT CONST laenge, von) --> TEXT
CAT ..... (TEXT VAR rechts, TEXT CONST links)
+ ..... (TEXT CONST links, rechts) --> TEXT
* ..... (INT CONST times, TEXT CONST quelle) --> TEXT
laenge ..... (TEXT CONST text) --> INT
LENGTH ..... (TEXT CONST text) --> INT
pos ..... (TEXT CONST quelle, muster) --> INT
pos ..... (TEXT CONST quelle, muster, INT CONST von) --> INT
pos ..... (TEXT CONST quelle, muster, INT CONST von, bis) --> INT
compress ..... (TEXT CONST text) --> TEXT
change ..... (TEXT VAR ziel, TEXT CONST altes muster, neues muster)
```

Datentyp BOOL

```
:= ..... (BOOL VAR ziel, BOOL CONST quelle)
NOT ..... (BOOL CONST operand) --> BOOL
AND ..... (BOOL CONST links, rechts) --> BOOL
OR ..... (BOOL CONST links, rechts) --> BOOL
CAND ..... (BOOL CONST links, rechts) --> BOOL
COR ..... (BOOL CONST links, rechts) --> BOOL
XOR ..... (BOOL CONST links, rechts) --> BOOL
```

Datentyp REAL

```
maxreal ....... --> REAL
smallreal ..... --> REAL
:= ............ (REAL VAR ziel, REAL CONST quelle)
+ ............. (REAL CONST links, rechts) --> REAL
- ............. (REAL CONST links, rechts) --> REAL
* ............. (REAL CONST links, rechts) --> REAL
/ ............. (REAL CONST links, rechts) --> REAL
- ............. (REAL CONST operand) --> REAL
= ............. (REAL CONST links, rechts) --> BOOL
<> ............ (REAL CONST links, rechts) --> BOOL
< ............. (REAL CONST links, rechts) --> BOOL
<= ............ (REAL CONST links, rechts) --> BOOL
> ............. (REAL CONST links, rechts) --> BOOL
>= ............ (REAL CONST links, rechts) --> BOOL
floor ......... (REAL CONST operand) --> REAL
abs ........... (REAL CONST operand) --> REAL
ABS ........... (REAL CONST operand) --> REAL
sign .......... (REAL CONST operand) --> INT
SIGN .......... (REAL CONST operand) --> INT
MOD ........... (REAL CONST links, rechts) --> REAL
max ........... (REAL CONST a, b) --> REAL
min ........... (REAL CONST a, b) --> REAL
INCR .......... (REAL VAR ziel, REAL CONST increment)
DECR .......... (REAL VAR ziel, REAL CONST decrement)

pi ............ --> REAL
e ............. --> REAL
ln ............ (REAL CONST x) --> REAL
log2 .......... (REAL CONST z) --> REAL
log10 ......... (REAL CONST x) --> REAL
sqrt .......... (REAL CONST z) --> REAL
exp ........... (REAL CONST z) --> REAL
tan ........... (REAL CONST x) --> REAL
tand .......... (REAL CONST x) --> REAL
sin ........... (REAL CONST x) --> REAL
sind .......... (REAL CONST x) --> REAL
```

```
cos ..... (REAL CONST x) --> REAL
cosd ..... (REAL CONST x) --> REAL
arctan ..... (REAL CONST x) --> REAL
arctand ..... (REAL CONST x) --> REAL
** ..... (REAL CONST links, INT CONST b) --> REAL
random ..... --> REAL
initializerandom (REAL CONST startwert)
```

Konversion von Werten

```
text ..... (INT CONST zahl) --> TEXT
text ..... (INT CONST zahl, laenge) --> TEXT
text ..... (REAL CONST zahl) --> TEXT
text ..... (REAL CONST zahl, INT CONST laenge, nachkommastellen) --> TEXT

int ..... (TEXT CONST zahl) --> INT
int ..... (REAL CONST zahl) --> INT

real ..... (INT CONST zahl) --> REAL
real ..... (TEXT CONST text) --> REAL

lastconversionok --> BOOL
```

Dialog-Ein/Ausgabe

```
put ..... (TEXT CONST text)
putline ..... (TEXT CONST text)
line .....
line ..... (INT CONST n)
page .....

get ..... (TEXT VAR wort)
get ..... (TEXT VAR wort, TEXT CONST separatorzeichen)
get ..... (TEXT VAR wort, INT CONST laenge)
getline ..... (TEXT VAR zeile)

put ..... (INT CONST zahl)
get ..... (INT VAR zahl)

put ..... (REAL CONST zahl)
get ..... (REAL VAR zahl)
```

Dateiverarbeitung

```
input ......... --> TRANSPUTDIRECTION
output ........ --> TRANSPUTDIRECTION
sequentialfile  (TRANSPUTDIRECTION CONST modus, TEXT CONST name) --> FILE
maxlinelaenge . (FILE CONST file) --> INT
maxlinelaenge . (FILE VAR file, INT CONST laenge)
putline ....... (FILE VAR file, TEXT CONST zeile)
getline ....... (FILE VAR file, TEXT VAR zeile)
line .......... (FILE VAR file)
reset ......... (FILE VAR file)
eof ........... (FILE CONST file) --> BOOL
line .......... (FILE VAR file, INT CONST n)
page .......... (FILE VAR file)
put ........... (FILE VAR file, TEXT CONST wort)
put ........... (FILE VAR f, INT CONST zahl)
put ........... (FILE VAR f, REAL CONST zahl)
get ........... (FILE VAR file, TEXT VAR wort, TEXT CONST separator)
get ........... (FILE VAR file, TEXT VAR wort, INT CONST maxlaenge)
get ........... (FILE VAR file, TEXT VAR wort)
get ........... (FILE VAR f, INT VAR zahl)
get ........... (FILE VAR f, REAL VAR zahl)
close ......... (FILE VAR file)
:= ............ (FILE VAR ziel, FILE CONST quelle)
```

COMPLEX

```
complexzero ... --> COMPLEX
complexone .... --> COMPLEX
complexi ...... --> COMPLEX
:= ............ (COMPLEX VAR ziel, COMPLEX CONST quelle)
complex ....... (REAL CONST re, im) --> COMPLEX
realpart ...... (COMPLEX CONST number) --> REAL
imagpart ...... (COMPLEX CONST number) --> REAL
CONJ .......... (COMPLEX CONST number) --> COMPLEX
= ............. (COMPLEX CONST links, rechts) --> BOOL
<> ............ (COMPLEX CONST links, rechts) --> BOOL
```

```
+ ..... (COMPLEX CONST links, rechts) --> COMPLEX
- ..... (COMPLEX CONST links, rechts) --> COMPLEX
* ..... (COMPLEX CONST links, rechts) --> COMPLEX
/ ..... (COMPLEX CONST links, rechts) --> COMPLEX
get ..... (COMPLEX VAR links)
put ..... (COMPLEX CONST links)
dphi ..... (COMPLEX CONST x) --> REAL
phi ..... (COMPLEX CONST x) --> REAL
dphi ..... (COMPLEX CONST x) --> REAL
phi ..... (COMPLEX CONST x) --> REAL
sqrt ..... (COMPLEX CONST x) --> COMPLEX
ABS ..... (COMPLEX CONST x) --> REAL
```

VECTOR

```
:= ..... (VECTOR VAR l, VECTOR CONST r)
vector ..... (INT CONST lng, REAL CONST value) --> VECTOR
vector ..... (INT CONST lng) --> VECTOR
SUB ..... (VECTOR CONST v, INT CONST i) --> REAL
LENGTH ..... (VECTOR CONST v) --> INT
laenge ..... (VECTOR CONST v) --> INT
norm ..... (VECTOR CONST v) --> REAL
replace ..... (VECTOR VAR v, INT CONST i, REAL CONST r)
= ..... (VECTOR CONST l, r) --> BOOL
<> ..... (VECTOR CONST l, r) --> BOOL
+ ..... (VECTOR CONST v) --> VECTOR
+ ..... (VECTOR CONST l, r) --> VECTOR
- ..... (VECTOR CONST links) --> VECTOR
- ..... (VECTOR CONST l, r) --> VECTOR
* ..... (VECTOR CONST l, r) --> REAL
* ..... (VECTOR CONST v, REAL CONST r) --> VECTOR
* ..... (REAL CONST r, VECTOR CONST links) --> VECTOR
/ ..... (VECTOR CONST links, REAL CONST r) --> VECTOR
get ..... (VECTOR VAR v, INT CONST lng)
put ..... (VECTOR CONST v, INT CONST laenge, fracs)
put ..... (VECTOR CONST v)
```

MATRIX

```
:= ..... (MATRIX VAR l, MATRIX CONST r)
matrix ..... (INT CONST rows, columns, REAL CONST value) --> MATRIX
matrix ..... (INT CONST rows, columns) --> MATRIX
idn ..... (INT CONST size) --> MATRIX
row ..... (MATRIX CONST m, INT CONST i) --> VECTOR
column ..... (MATRIX CONST m, INT CONST j) --> VECTOR
COLUMNS ..... (MATRIX CONST m) --> INT
ROWS ..... (MATRIX CONST m) --> INT
sub ..... (MATRIX CONST links, INT CONST row, column) --> REAL
replacerow .... (MATRIX VAR m, INT CONST rowindex, VECTOR CONST rowvalue)
replacecolumn . (MATRIX VAR m,INT CONST columnindex, VECTOR CONST columnvalue)
replaceelement  (MATRIX VAR links, INT CONST row, column, REAL CONST x)
= ..... (MATRIX CONST l, r) --> BOOL
<> ..... (MATRIX CONST l, r) --> BOOL
+ ..... (MATRIX CONST m) --> MATRIX
+ ..... (MATRIX CONST l, r) --> MATRIX
- ..... (MATRIX CONST m) --> MATRIX
- ..... (MATRIX CONST l, r) --> MATRIX
* ..... (REAL CONST x, MATRIX CONST m) --> MATRIX
* ..... (MATRIX CONST m, REAL CONST x) --> MATRIX
* ..... (VECTOR CONST v, MATRIX CONST m) --> VECTOR
* ..... (MATRIX CONST m, VECTOR CONST v) --> VECTOR
* ..... (MATRIX CONST l, r) --> MATRIX
get ..... (MATRIX VAR links, INT CONST rows, columns)
put ..... (MATRIX CONST links, INT CONST laenge, fracs)
put ..... (MATRIX CONST links)
TRANSP ..... (MATRIX CONST m) --> MATRIX
transp ..... (MATRIX VAR m)
INV ..... (MATRIX CONST m) --> MATRIX
DET ..... (MATRIX CONST m) --> REAL
```

Fehlermeldungen

```
errorsstop ..... (TEXT CONST meldung)
stop .....
```

Anhang B

```
PACKET stellenaddition DEFINES
                       erfrage zwei positive zahlen ,
                       melde die summe als ergebnis ,
                       setze uebertrag auf null ,
                       beruecksichtige letzten uebertrag ,
                       nimm einerstelle als aktuelle stelle ,
                       addiere aktuelle stelle mit uebertrag ,
                       nimm naechste als aktuelle stelle ,
                       noch stellen zu addieren :

     TEXT VAR zahl1, zahl2, summe ;
     INT VAR  stelle, uebertrag ;

PROC erfrage zwei positive zahlen :
     summe := "" ;
     erfrage (zahl1) ;
     erfrage (zahl2) ;
     bringe auf gleiche stellenzahl .

bringe auf gleiche stellenzahl :
     INT CONST stellendifferenz :: LENGTH zahl1 - LENGTH zahl2 ;
     IF stellendifferenz >= 0
       THEN zahl2 :=  stellendifferenz * "0" + zahl2
       ELSE zahl1 := -stellendifferenz * "0" + zahl1
     FI .

ENDPROC erfrage zwei positive zahlen ;

PROC erfrage (TEXT VAR zahl) :
     line ;
     put ("Bitte geben Sie eine natürliche Zahl ein:") ;
     getline (zahl) ;
     IF nicht nur ziffern
       THEN put ("So nicht! Das ist keine natürliche Zahl!") ;
            erfrage (zahl)
     FI .

nicht nur ziffern :
     INT VAR i ;
     FOR i FROM 1 UPTO LENGTH zahl REP
          IF ist keine ziffer
            THEN LEAVE nicht nur ziffern WITH TRUE
          FI
     ENDREP ;
     FALSE .

ist keine ziffer :  pos ("0123456789", zahl SUB i) = 0 .
ENDPROC erfrage ;
```

```
PROC melde die summe als ergebnis :
     put ("Ergebnis:") ;
     put (summe)
ENDPROC melde die summe als ergebnis ;

PROC setze uebertrag auf null :
     uebertrag := 0
ENDPROC setze uebertrag auf null ;

PROC nimm einerstelle als aktuelle stelle :
     stelle := LENGTH zahl1
ENDPROC nimm einerstelle als aktuelle stelle ;

PROC nimm naechste als aktuelle stelle :
     stelle DECR 1
ENDPROC nimm naechste als aktuelle stelle ;

BOOL PROC noch stellen zu addieren :
     stelle > 0
ENDPROC noch stellen zu addieren ;

PROC addiere aktuelle stelle mit uebertrag :
     INT VAR stellensumme :=
     int (zahl1 SUB stelle) + int (zahl2 SUB stelle) + uebertrag ;
     IF stellensumme <= 9
       THEN uebertrag := 0
       ELSE uebertrag := 1 ;
            stellensumme DECR 10
     FI ;
     summe := text (stellensumme) + summe
ENDPROC addiere aktuelle stelle mit uebertrag ;

PROC beruecksichtige letzten uebertrag :
     IF uebertrag = 1
       THEN summe := "1" + summe
     FI
ENDPROC beruecksichtige letzten uebertrag ;

ENDPACKET stellenaddition ;
```

Literaturverzeichnis

1) E.W.Dijkstra,
A Discipline of Programming, Englewood Cliffs, N.J., 1976

2) A.Engel,
Elementarmathematik vom algorithmischen Standpunkt, Stuttgart 1972

3) R.Hahn,
Höhere Programmiersprachen im Vergleich, Wiesbaden 1981

4) R.Hahn,
Erste Hilfe in ELAN, Teil 1 und Teil 2, HRZ Bielefeld, 1983

5) R.Hahn, D.Heinrichs, P.Heyderhoff, J.Liedtke,
EUMEL-Benutzerhandbuch, GMD Bonn-Birlinghoven u. HRZ Bielefeld, 1982

6) R.Hahn, B.Nienaber,
Probleme lösen mit dem Computer, Teil 1 und Teil 2, Tübingen 1976

7) R.Hahn, P.Stock,
ELAN-Handbuch, Wiesbaden 1979

8) G.Hommel, J.Jäckel, St.Jähnichen, K.Kleine, W.Koch, C.H.A.Koster,
ELAN-Sprachbeschreibung, Wiesbaden 1979

9) G.Hommel, St.Jähnichen, C.H.A.Koster,
Methodisches Programmieren, Entwicklung von Algorithmen durch schrittweise Verfeinerung, Berlin u. New York 1983

10) E.Horowitz,S.Sahni,
Algorithmen, Berlin u. Heidelberg u.New York, 1981

11) L.H.Klingen,
Elementare Algorithmen, Prozeduren und Probleme, Freiburg u. Basel u. Wien, 1981

12) D.E.Knuth,
The Art of Computer Programming, Reading, 1968

13) N.Wirth,
Systematisches Programmieren, Stuttgart, 1983

14) N.Wirth,
Algorithmen und Datenstrukturen, Stuttgart, 1983

Sachregister

Abbruchbedingung 24
Abschnitt 10
abstrakte Typen 160
Abstraktionsebene 60
Accessattribut 28
aktuelle Parameterliste 81
Algorithmus 8
AND 17
Anweisungen 10
ASCII-Code 153
Aufruf 76
Ausgleichskurve 105
Auswahl 16
Backtracking 131
Basisoperationen 13, 17
Bedingung 12, 16
Bezeichner 12
binäre Suche 115
Binomialkoeffizienten 111
Bisektion 112, 48
BOOL 26
Bruchrechnung 160
Bruttosozialprodukt 107
CAND 57
CONCR 165
CONST 28
COR 57
Datenbank 122
Datendeklaration 29
Datenstruktur 123
Datentypen 16, 26
Datenverbunde 124
DECR 21
DEFINES-Liste 147
Deklaration 20
Deklarationen im Paket 148
Dekrementieren 21
Denoter 18
Designentscheidungen 59
DOWNTO 56
dynamische Daten 148
Ein- und Ausgangsparameter 81
Eingangsparameter 81
Einheiten für begreifendes Handeln 141
ELAN-Maschine 149
ELSE 16
eof 33
errorstop 85
Euklidische Algorithmus 110
Euklidischer Algorithmus 160
Exponent 43
Fachsprache 141
FALSE 26
Feinstruktur 164
FI 29
Fibonacci-Folge 110
FILE 33
Filter 60
FOR 56
formale Parameterliste 81
FROM 56
Gaußelimination 100
Generische Prozeduren 91
geometrische Reihe 162
get 18
getline 23
ggt 110
Gierige Algorithmen 93
globale Variable 76
Hash mit Rehash 158
Hash mit Verkettung 158
Hashfunktion 152
Hashing 152
Hashklassen 152
Hashtabelle 155
Hashverfahren 152
Heronsche Methode 50
Horner-Verfahren 168
IF-Auswahl 16
INCR 21
Index 55
indirekte Rekursionen 109
Initialisierung 29
Inkrementieren 21
INT 19
INV 105
Invarianzaussage 54
kgV 110
Konkretisierer 165
Konstruktor 165
Korrekturhilfe 142, 143
Labyrinth 131
Laufvariable 56
LEAVE 52
LEAVE-Anweisung 92
LENGTH 20
LET 125, 58
Lexikon behandeln 143
lineare Gleichungssysteme 100
lineare Suche 112
Lokale Objekte 76
Mantisse 43
matrix 105
Maus sucht Käse 131
Maximumsuche 31
maxint 134
Median 122
Minimalgerüst 99
Mischen 116
MOD 53

Modularisierung 148
Monadische Operatoren 27
Muster suchen 145
Namen 12
NOT 34
Nullstellen 183
Operatorbezeichner 27
Operatoren 163, 163, 27
Operatornamen 163
Operatorrumpf 164
Paket 145, 147
Paket- und Prozedurdaten 148
Paketbibliothek 150
Paketrumpf 147
Palindrom 15
Parameter 81, 18
Parameterübergabe 81, 90
Pascal-Dreieck 111
Pivot 117
Polynom 167
Polynomdivision 170
Pseudodivison auf Pol. 181
Primzahl 51
Prioritätenregelung 28
Programmentwicklung 59
Prozedur 76, 73, 86
Prozeduren als Parameter 97
Prozedurkopf 81
Prozedurrumpf 76
Punkt 10
put 18
Quasizufallszahl 34
Quicksort 119
random 34
REAL 42
Refinement 85, 12
Regression 105
Rehash-Funktion 158
Reihung 55
Rekursion 146
Rekursive Algorithmen 106
REP 29
replace 74
Rösselsprung 141
ROW 55
Rucksack 98, 141
Rundungsfehler 45
Samuelson-Modell 109
scanner 151
Schleife 12
Schleifenrumpf 12
Schlüsselworte 12
Seiteneffekt 62
Seitenwirkungen 87
Selektion 124
Semikolon 9
sequential file 33
Sieb des Erathostenes 54
Silbentrennung 144
Sortieren 112
Splines 104
Standardpaketen 150
Standardprozeduren 91
statische Daten 148
STRUCT 124
Strukturen 124
Strukturierung 66
SUB 21
Subskription 55
Suchen 112
TEXT 18
THEN 16
Top-Down 83
Topdown-Entwicklung 61
Travelling Salesman Problem 141
TRUE 26
Türme von Hanoi 110
UNTIL 24
UPTO 56
VAR 28
Variable 18
vector 105
Verbalisierung 65
Verkürzte Schlüsselworte 29
Verweisliste 126
virtuelle Parameterliste 97
vorübersetzen 150
Wertliefernde Ausdrücke 28
Wertliefernde IF-Auswahl 50
Wertliefernde Prozeduren 89
Wertliefernde Refinements 39
Wertzuweisung 28
Wheatstone-Brücke 104
WHILE-Schleife 12
Wiederholung 11
WITH 92
Wortweises Lesen 143
Wurzel 42
Zeichenweises Lesen 144
Zeiger 155
Zugriffsrecht 28
Zuweisung 164, 20

MikroComputer–Praxis

Die Teubner-Buchreihe für Ausbildung, Freizeit und Hobby

Erbs/Stolz: **Einführung in die Programmierung mit PASCAL**
232 Seiten. DM 22,80

Haase/Stucky/Wegner: **Datenverarbeitung heute**
284 Seiten. DM 21,80

Hainer: **Numerik mit BASIC-Tischrechnern**
In Vorbereitung

Klingen/Liedtke: **Programmieren mit ELAN**
207 Seiten. DM 22,80

Lehmann: **Lineare Algebra mit dem Computer**
285 Seiten. DM 23,80

Löthe/Quehl: **Systematisches Arbeiten mit BASIC**
188 Seiten. DM 19,80

Menzel: **BASIC in 100 Beispielen**
3. Aufl. 214 Seiten. DM 22,80
– **mit Diskette:** Alle BASIC-Programme in APPLESOFT
DM 62,–

Menzel: **Dateiverarbeitung mit BASIC**
In Vorbereitung
– **mit Diskette:** Alle BASIC-Programme in CP/M-Version und APPLE-DOS 3.3-Version sowie eine Testdatei
In Vorbereitung

Nievergelt/Ventura: **Die Gestaltung interaktiver Programme**
124 Seiten. DM 21,80
– **mit Diskette:** UCSD-Pascal-Programme für den Apple II Computer
DM 59,80

Ottmann/Schrapp/Widmayer: **PASCAL in 100 Beispielen**
In Vorbereitung
– **mit Diskette:** UCSD-Pascal-Programme für den Apple II Computer
In Vorbereitung

Die Reihe wird durch weitere Bände fortgesetzt.

Preisänderungen vorbehalten

B. G. Teubner Stuttgart