SYMBOL

Hinweis Ab 15.6. 1966 gilt:

Unmittelbar hinter dem letzten Befehl müssen 5 Malteserkreuz-Zeichen stehen (✠)

er56 - adressierprogramm s y m b o l

von a.schoenhage
universitaet koeln
rechenzentrum

1. allgemeines

s y m b o l uebersetzt programme mit symbolischer adressierung und einigen anderen bequemen notationen in expl'zite maschinenprogramme. einzelobjekte der uebersetzung sind (im unterschied zu einem compiler fuer reine formelsprache) befehle, programmkonstanten und weitere der uebersetzung dienende hinwei:e. bei befehlen ist die operation explizit, also durch zwei ziffern anzugeben, fuer adress- und indexteil kann eine flexible symbolische schreibweise benutzt werden. als programmkonstanten lassen sich in bequemer form gleitkommazahlen und alphatext eingeben.

die praktische handhabung von s y m b o l sieht etwa so aus:
ablochen des programmtextes am fernschreiber,
2aus5-wandlung in alphatext,
s y m b o l einlesen,
alphastreifen am fotoleser einlegen und s y m b o l starten.
bei sprachlichen fehlern stanzt s y m b o l entsprechende hinweise in alphatext aus und stopt.
bei korrektem text wird das uebersetzte pogramm auf lochstreifen ausgestanzt.
auf wunsch stanzt s y m b o l weiter (in alphatext) ein verzeichnis der symbolischen adressen als unterlage fuer die weitere programmbenutzung aus.

im folgenden werden die syntaktischen regeln beschrieben, wozu metasprachliche formeln benutzt werden (vgl. algol 60), z.b.

(vorzeichen)::- + v -

die in klammern gesetzte bezeichnung eines sprachlichen bausteins steht stellvertretend fuer einen beliebigen solchen baustein.
die zeichenfolge ::- ist zu lesen als 'kann sein'. v ist als logisches 'oder' gemeint.
die im folgenden benutzten symbole des er56-alpha-codes lassen sich ohne weiteres durch andere ersetzen. s y m b o l ist dann nur entsprechend abzuaendern.

2. syntaktische regeln

2.1. benutzte symbole

(ziffer) ::- 0 v 1 v ... v 9 (00-09 intern)
(zeichen)::- + v - v , v . v □ v : v = v ' v (v)
(11 40 22 23 20 12 10 13 14 15 intern)
(buchstabe) ::- a v b v ... v y v z (60-89 intern)
(trennung) ::- wr v zv v zw (42 41 39 intern)
als symbol ganz allgemein werden alle internen verschluesselungen 00-99 verstanden (omega wird an der vorzeichendezimale 9 erkannt).

2.2. syntaktische einheiten

(einheit)::=(befehl)v(zahl)v(text)v(kommentar)v(definition)
v(bezugspunkt)v(omegaende)v(trennung)(einheit)

hier sind die einzelnen bausteine aufgefuehrt, aus denen sich ein programm zusammensetzen kann. das ende bilden zwei omegaworte - oben als omegaende bezeichnet, also rekursiv

(programm)::= (omegaende) v (einheit)(programm)

s y m b o l entschluesselt zyklisch einheit um einheit. am anfang einer einheit kann nach obigem eine beliebige zahl von trennungen stehen. als ende dieser einheit (und gleichzeitig als anfang der naechsten) wirkt das erste symbol, das nach den syntaktischen regeln keine gueltige fortsetzung der alten einheit sein kann. dabei haben trennungen oft eine wichtige bedeutung, z.b. bei zahlen (vgl. 2.9.): -3.45 ist eine einheit, dagegen stellt -3 .45 zwei einheiten dar (zwischenraum als trennung).

2.3. namen

(name) ::= (buchstabe) v (name)(buchstabe) v (name)(ziffer)
v (trennung)(name)

also jede symbolfolge, die nur buchstaben und ziffern enthaelt und mit einem buchstaben beginnt, z.b. cis a u34z5. namen dienen zur symbolischen bezeichnung von adressgroessen. wie sie als solche festgelegt werden, wird in 2.7. und 2.8. erlaeutert.

2.4. adressen

unabhaengig von der syntaktischen darstellung enthaelt die bedeutung einer adresse stets neben den vier adressstellen auch eine indexstelle. dabei werden unterschieden:

xxxx 0 absolute adresse
xxxx 1 (i ungleich null) indizierte adresse
und darin speziell
xxxx 9 relative adresse

hier dient der index 9 zunaechst nur als indikator, waehrend der adressteil die relative adresse bezueglich programmanfang enthaelt. erst bei einsetzen in einen befehl wird die notwendige umrechnung auf die spezielle position dieses befehls vorgenommen.

2.4.1. explizite adressen

(expladr) ::= (ziffer)(ziffer)(ziffer)(ziffer)
v(ziffer)(ziffer)(ziffer)
v(ziffer)(ziffer)
v(ziffer) v (trennung)(expladr)

obige definition ist so zu verstehen, dass moeglichst viele, aber hoechstens vier aufeinanderfolgende ziffern als absolute adresse gelesen werden. bei weniger als vier ziffern werden vorangehende nullen ergaenzt.

2.4.2. adressummanden

(summand) ::= (vorzeichen)(name) v (vorzeichen)(expladr)
v (trennung)(summand)

2.4.3. adressen

(adresse) ::= (name) v (expladr) v (adresse)(summand)

diese allgemeine form einer adresse erlaubt eine flexible anwendung der symbolischen adressierung. so kann z.b. das zweite wort einer unter adresse c17u gespeicherten gleitkommazahl mittels c17u + 1 addressiert werden.
die addition bzw. subtraktion von adressummanden erfolgt in rechnung mit 5-stelligen zahlen modulo 100000 , also z.b.
0000 0 - 0200 9 = 9799 1 .

2.5. index, operation

(index) ::= (ziffer) v (trennung)(index)
(operation) ::= (ziffer)(ziffer) v (trennung)(operation)

(iop) ::= (index)(operation) v (operation)

moegliche schreibweisen sind demnach z.b. xxx oder x xx oder xx mit hier notwendig folgendem symbol, das keine ziffer ist, weil sonst die erste lesart moeglich waere. bei fehlendem index wird 0 ergaenzt.

2.6. befehle

(befehl) ::= (adresse)(iop)

bei befehlen enthalten also formal die adresse und auch iop eine indexstelle. s y m b o l prueft deren vertraeglichkeit nach folgenden regeln:

2.6.1. bei index 0 und operationen 00-89 sind beliebige adressen moeglich.

2.6.2. bei index verschieden von null oder operationen 90-99 sind nur absolute adressen moeglich, abgesehen von folgender ausnahme:

2.6.3. bei iop 993 oder 995 sind absolute oder relative adressen moeglich. absolute adressen werden einfach eingesetzt. bei relativen adressen dagegen wird neben der notwendigen umrechnung auf die spezielle position des befehls im falle 995 noch auf 10000 komplementiert. demnach wirken 2 9 93 und 2 995 verschieden, 2 9 93 und 99989 95 gleich. dagegen wirken (wenn test eine relative adresse bezeichnet) test 993 und test 995 gleich.

wie bei der trennung von einheiten sind auch zwischen den einzelnen bausteinen eines befehls oft trennungen noetig. es lassen sich gemaess der obigen festsetzungen beliebi e folgen von je 7 ziffern ohne trennung als explizite maschinenworte eingeben.

2.7. bezugspunkte

(doppelpunkt) ::= : v (trennung)(doppelpunkt)

(bezugspunkt) ::= (name)(doppelpunkt)

s y m b o l beginnt das programm mit relativadresse 0000 9, die nach jedem gelesenen befehl um 0001 0 erhöht wird (bei programmkonstanten entsprechend mehr, vgl. unten). mit dem nachfolgenden doppelpunkt wird der aufgeführte name durch den augenblicklichen zählerstand als relative adresse definiert.
es können mehrere bezugspunkte nacheinander stehen, sie bezeichnen dann die gleiche relativadresse.

2.8. adressendefinitionen

(namenliste) ::= (name) v (namenliste)(trennung)
v (namenliste),(name)

(definition) ::= (namenliste)=(adresse)(index)

während durch bezugspunkte nur relative adressen innerhalb des programms definiert werden, lassen sich hier beliebige adressen definieren. der index wird stets als positiver summand

0000 (index)

interpretiert. er darf aus syntaktischen gründen auch dann nicht fehlen, wenn er 0 lautet.
der sinn einer einfachen definition

(name)=(adresse)(index)

ist offensichtlich. bei mehreren namen in der liste wirkt jedes komma als eine adressenerhöhung um 0002 ; . a, c17u ,rd = 0 1
bedeutet also ausführlich a =00001 c17u= a+2 0rd =c17u+ 2 0
(die letzte zeile wurde absichtlich etwas unübersichtlich geschrieben, um zu verdeutlichen, wo trennungen stehen müssen, stehen können und fehlen dürfen).
bei mehrfacher definition eines namens stopt s y m b o l. auch a=a 0 a=0 0 oder xz:xz: z.b. gelten als doppelte definition.

2.9. zahlen

(mantissenvorzeichen)::= + v - v ++ v --
(dezimalbruch)::= .(ziffer) v (dezimalbruch)(ziffer)
(dezimalzahl)::=(ziffer)v(dezimalbruch)v(ziffer)(dezimalzahl)
(mantisse)::=(dezimalbruch)v(mantissenvorzeichen)(dezimalzahl)
(exponent)::=(ziffer)v(ziffer)(ziffer)v(vorzeichen)(ziffer)
v(vorzeichen)(ziffer)(ziffer)

(zahl)::=(mantisse)⌐(exponent)v(mantisse)v⌐(exponent)
v(mantissenvorzeichen)⌐(exponent)

nach den hier angegebenen regeln kann eine zahl (zur unterscheidung von expliziten adressen) nicht mit einer ziffer beginnen. weiter darf eine zahl keine trennung enthalten.
beispiele: +1 --1.2 .045 ++00.03⌐4 +2⌐47 ++⌐+3 ⌐-08
fehlende vorzeichen werden durch + ergänzt. fehlende mantisse wird durch 1.0 ergänzt. fehlender exponent wird durch ⌐+00 ergänzt. für die interne darstellung ist zu beachten:
durch ++ bzw. -- erhält die zahl ein q-zeichen. zahlen mit mehr als 11 signifikanten ziffern (vorlaufende nullen werden nicht mitgezählt) sind unzulässig, sonst syntaxfehler. zahlen werden stets in ihre normalisierte gleitkommaform übersetzt. bei zahlen, die ungleich null sind und betraglich nicht in den bereich .1⌐-49 .99999999999⌐+49 fallen, erfolgt stop. bei mantisse null wird charakteristik 00 eingesetzt, unabhängig von einem evtl. geschriebenen exponenten.

2.10. text

(text)::= (symbol)(symbolfolge, die kein ' enthaelt)'

hiermit kann beliebiger alphatext als programmkonstante eingegeben werden, wobei ' am anfang und am ende natuerlich nicht mit uebernommen werden. das letzte wort erhaelt ein q-zeichen, also vorzeichendezimale 6, und wird gegebenenfalls mit dem symbol 43 (umschaltung auf kleinschreibung) aufgefuellt.
nach obigem wird das erste symbol nicht auf textende geprueft. d.h. es gibt keinen leeren text. deshalb kann man nun auch das apostroph selbst eingeben, indem man '' setzt, z.b. tom's car als 'tom''s car'(allerdings ist zu beachten, dass dann der text auch vor seinem ende q-zeichen enthaelt).

2.11. kommentare

kommentare sind in klammern gesetzte bemerkungen, die nicht uebersetzt werden. diese koennen ihrerseits wieder eingeklammertes enthalten. s y m b o l zaehlt jede klammeroeffnung als +1, jede klammerschliessung als -1, und ein kommentar ist beendet, wenn diese zaehlung wieder 0 erreicht, also

(kommentarende) ::=)
(kommentaranfang) ::= (v (kommentaranfang)(symbol ungleich ()) v (kommentaranfang)(kommentar)
(kommentar) ::= (kommentaranfang)(kommentarende)

beispiel: 1 5 93 (erhoehung von (15))
kommentare werden nach der augenblicklichen fassung von s y m b o l voellig ueberlesen. es ist geplant, bei vorhandenem schnelldrucker eine ausfuehrliche niederschrift des uebersetzten programms zu liefern, worin dann auch die eingegebenen kommentare unter fortlassung der aussensten einklammerung erscheinen.

3. allgemeine programmbeschreibung zu s y m b o l

3.1. speichereinteilung

0000-0999: speicherraum fuer das uebersetzte programm.
(15) zeigt lfd. an, welche zelle als naechste zu belegen ist, wird also am anfang auf 0000 gesetzt und enthaelt am ende der uebersetzung die adresse des ersten omegawortes.
ein programm im hier verstandenen sinne soll sich durch e i n e n befehl vom band oder vom lochstreifen einlesen lassen und kann demnach hoechstens einen teilspeicher belegen.

1000-1690: s y m b o l , start auf 1000.

1691-2899: namen- und summenspeicher.
ab 1691 aufsteigend werden alle vorkommenden namen mit ihrer bedeutung gespeichert, lfd. index n=(13).
ab 2899 absteigend werden fuer adressummanden kennwoerter gespeichert, lfd. index s=(14)

2900-2999: eingabespeicher 2.

3900-3999: eingabespeicher 1.
die speicheruebertaufadresse 1. 3000-speicher ist auf 3999 einzustellen (normalstellung). ab 3900 werden jeweils 100 worte des programmtextes eingelesen, nach 2900 umgespeichert und dort entschluesselt, waehrend schon wieder nach 3900 gelesen wird.

3000-3899: nicht belegt.

3.2. moegliche fehlerstops

bei stop 9002 0 19 stanzt s y m b o l einen der folgenden hinweise aus, wobei evtl. adressangaben nach moeglichkeit relativ zum zuletzt entschluesselten bezugspunkt gemacht werden, also z.b. anton+004: ...

(name)+....: omega im letzten text
(name)+....: omega im letzten kommentar
.... 9: iop mit adresse unvereinbar
programm zu lang (wenn zelle 0998 gefuellt wird)
zu viele namen (wenn 1691-2899 nicht ausreicht)
(name)+....: syntax-fehler
(name)+....: doppelte definition
(name)+....: gleitkomma-bereich verletzt
(name)+....: kein alpha bei eingabe (wenn ein wort des eingabetextes nicht vorzeichendezimale 3, 6 oder 9 hat)

(name) nicht definier.

diese letzte fehlerart wird erst moniert, wenn das lesen beendet ist. hier bewirkt druecken der starttaste, dass s y m b o l das gelesene omegaende ignoriert und eine fortsetzung des programmtextes lesen will. indem man am fotoleser z.b. die fehlende definition einlegt, lassen sich fehler dieser art sofort beheben. bei allen anderen fehlern ist keine direkte korrektur moeglich, sondern voellige neuuebersetzung erforderlich.

3.3. allgemeiner ablauf

start s y m b o l auf 1000.

stop 9002 0 19 nach ausgabe der fehler-ursache.

stop 9001 0 19 nach ausgabe des uebersetzten programms.

bei betaetigen der starttaste erfolgt ausgabe des namenverzeichnisses und

stop 9003 0 19.

bei betaetigen der starttaste beginn einer neuen uebersetzung.

th stuttgart
rechenInstitut

symbol-kurs
ss 1963

dipl. math. p. nopp

(beispiel 1. symbol-programm fuer er 56 zur berechnung der
summe s = x[1]+...+x[i]+...+x[n].
programm: ab zelle 0000.
betriebsart: gleitkomma.
daten: n = (2002*). x[i], i=1,...,n, in aufeinander-
folgenden doppelzellen gespeichert;
x[1] = (2004,2005); 0 in hilfszelle am
programmende.
ergebnis: s = (2000,2001).)

(pos)	(bef)		(bem)
(0000)	2000 0 19		(betriebsart gleitkomma)
	2002 1 90		
	2002 1 92		
	2003 1 96		(bereitstellen der zahl 2*n in der zelle 2003*)
	0012 0 31		(s:=0)
(0005)	0000 1 91		(i:=1)
	2004 1 35		(s:=s+x[i])
	0002 1 93		(i:=i+1)
	2003 1 97		(i vgl n+1 bzw. (j1) vgl 2*n)
	0006 0 12		(sprung, wenn ungleich)
(0010)	2000 0 32		(resultat nach zelle 2000 abspeichern)
	9000 0 19		(stop)
	1000 0 00		(konstante 0)
	0000 0 00		
	9999 9 99		
	9999 9 99		

fernschreibprotokoll der uebersetzung und des namenverzeichnisses

2000019
2002190
2002192
2003196
0012031
0000191
2004135
0002193
2003197
0006012
2000032
9000019
1000000
0000000
9999999
9999999

namenverzeichnis:

beispiel 2. symbol-programm fuer er 56 zur berechnung der
summe s = x[1]+...+x[i]+...+x[n].
programm relativ.
betriebsart: gleitkomma.
daten: n = (2002*); x[i], i = 1,...,n, in aufeinanderfolgenden doppelzellen gespeichert; x[1] = (2004, 2005); 0 in hilfszelle am programmende.
ergebnis: s = (2000, 2001).)

```
s,n,x1 = 2000 0 (adressendefinition)
2000 19          (fehlender index wird durch 0 ergaenzt)
n 1 90           (symbolische adresse n)
n 1 92
n+1 1 96         (symbolische adresse n+1, adressummand +1)
null 31          (symbolische adresse null durch bezugspunkt
                  definiert, wird in relative adresse uebersetzt)
0 1 91
pos: x1 1 35
2-1 93
n+1 1 97
pos 12           (symb. adre. pos durch bezugspunkt definiert, wird
                  in relative adresse uebersetzt)
s 32             (symb. adre., durch obige adressendefinition er-
                  klaert, wird in absolute adresse uebersetzt)
9000 19
null:+0          (eine zahl wird in 14ziffrige gk-form uebersetzt
                  und auf ihrer programmposition wieder abgesetzt.
                  eine zahl kann nicht mit einer ziffer beginnen.
                  syntax 2.9.)
9999999
9999999
```

fernschreibprotokoll der uebersetzung und des namenverezeichnisses

2000019
2002190
2002192
2003196
0007931
0000191
2004135
0002193
2003197
9996912
2000032
9000019
1000000
0000000
9999999
9999999

namenverzeichnis:

2000 0 - s
2002 0 - n
2004 0 - x1
0012 9 - null
0006 9 - pos

(beispiel 3. symbol-programm fuer er 56. aufgabe siehe
beispiel 2. hier raumsparende textgestaltung.)

```
        s,n,x1 = 2000 0 (ergebnis s, indexgrenze n, summand x1
start) 2000 19  n 1 90  n 1 92  n+1 1 96  null 31  0 1 91
pos:   x1 1 35  2 1 93  n+1 1 97  pos 12 (summationsschleife)
       s 32  9000 19 (stop)
null:  +0
       99999999999999
```

fernschreibprotokoll der uebersetzung und des namenverzeichnisses

```
2000019
2002190
2002192
2003196
0007931
0000191
2004135
0002193
2003197
9996912
2000032
9000019
1000000
0000000
9999999
9999999
```

namenverzeichnis:

```
2000 0 = s
2002 0 = n
2004 0 = x1
0012 9 = null
0006 9 = pos
```

(beispiel 4. symbol-programm fuer er 56. aufgabe siehe beispiel 2. hier andere textgestaltung, willkuerliche anordnung der adressendefinition.)

```
      2000 19
      n 190n 1 92 n+1 196 n-2002 0
      null 31 0 1 91
pos:  x1 1 35  2 1 93 n + 1 1 97 pos
               12

      s 32

      9000 19

null:+0  s- 20000  x1-20040

      9999999 9999999
```

fernschreibprotokoll der uebersetzung und des namenverzeichnisses

```
2000019
2002190
2002192
2003196
0007931
0000191
2004135
0002193
2003197
9996912
2000032
9000019
1000000
0000000
9999999
9999999
```

namenverzeichnis:

```
2002 0 = n
0012 9 = null
0006 9 = pos
2004 0 = x1
2000 0 = s
```

(beispiel 5. symbol-programm fuer er 5t. aufgabe siehe beispiel 2.
andere adressendefinitionen.)

```
      n = 2002 0  s = 2000 0  x1 = 2004 1
      2000 19
      n 1 90
      n 1 92
      n+1 1 96
      null 31
      0 1 91
pos:  x1 35
      2 1 93
      n+1 1 97
      pos 12
       s 32
      9000 19
null: +0
      9999999
      9999999
```

fernschreibprotokoll der uebersetzung und des namenverzeichnisses

```
2000019
2002190
2002192
2003196
0007931
0000191
2004135
0002193
2003197
9996912
2000032
9000019
1000000
0000000
9999999
9999999
```

namenverzeichnis:

```
2002 0 = n
2000 0 = s
2004 1 = x1
0012 9 = null
0006 9 = pos
```

```
(beispiel 6. symbol-programm fuer er 56. aufgabe siehe
              beispiel 2. lokalisiere den fehler.)
s.o.x1 = 2000 0 (adressendefinition)
200019          (fehlender index wird durch 0 ergaenzt)
n 1 90          (symbolische adresse n)
n 1 92
n+1 1 96        (symbolische adresse n+1, adressummand +1)
null 31         (symbolische adresse null durch bezugspunkt
                definiert, wird in relative adresse uebersetzt)
0 1 91
pos: x1 1 35
2 1 93
n+1 1 97
pos 12          (symb. adre. pos durch bezugspunkt definiert, wird
                in relative adresse uebersetzt)
s 32            (symb. adre., durch obige adressendefinition er-
                klaert, wird in absolute adresse uebersetzt)
9000 19
null: 0         (eine zahl wird in 14ziffrige gk-form uebersetzt
                und auf ihrer programmposition wieder abgesetzt.
                eine zahl kann nicht mit einer ziffer beginnen,
                syntax 2.9.)
9999999
9999999
```

ausgabe nach fehlerstop und reaktion des er 56

```
null+000: syntax-fehler
ich will nicht
```

```
beispiel 7. symbol-programm fuer er 56. aufgabe siehe
            beispiel 2. lokalisiere den fehler.

s,n,x1 = 2000 0 (adressendefinition)

2000 19          (fehlender index wird durch 0 ersetzt)

n 1 90           (symbolische adresse n)

n 1 92

n+1 1 96         (symbolische adresse n+1, adressbefehl +1)

null 31          (symbolische adresse null durch bezugspunkt
                 definiert, wird in relative adresse uebersetzt)

0 1 91

pos: x11 35

2 1 93

n+1 1 97

pos 12           (symb. adre. pos durch bezugspunkt definiert, wird
                 in relative adresse uebersetzt)

s 32             (symb. adre., durch obige adressendefinition er-
                 klaert, wird in absolute adresse uebersetzt)

9000 19

null:+0          (eine zahl wird in 1 ziffrige gk-form uebersetzt
                 und auf ihrer programmposition wieder abgesetzt.
                 eine zahl kann nicht mit einer ziffer beginnen,
                 mutex 2.9.)

9999999

9999999


ausgabe nach fehlerstop

x11 nicht definiert
```

```
(beispiel 8. symbol-programm fuer er 56. aufgabe siehe
          beispiel 2. lokalisiere den fehler.)
s,n,x1 = 2000 0 (adressendefinition)
2000 19          (fehlender index wird durch 0 ergaenzt)
n 1 90           (symbolische adresse n)
n 1 92
n+1 1 96         (symbolische adresse n+1, adressummand +1)
null 31          (symbolische adresse null durch bezugspunkt
                  definiert, wird in relative adresse uebersetzt)
0 1 91
pos: x1 1 35
2 1 93
n+1 1 97
pos912           (symb. adre. pos durch bezugspunkt definiert, wird
                  in relative adresse uebersetzt)
s 32             (symb. adre., durch obige adressendefinition er-
                  klaert, wird in absolute adresse uebersetzt)
9000 19
null:+0          (eine zahl wird in 14ziffrige gk-form uebersetzt
                  und auf ihrer programmposition wieder abgesetzt.
                  eine zahl kann nicht mit einer ziffer beginnen,
                  syntax 2.9.)
9999999
9999999

ausgabe nach fehlerstop

pos+003: syntax-fehler
```

```
(beispiel 9. symbol-programm fuer er 56. aufgabe siehe
             beispiel 2. lokalisiere den fehler.)

s,n,x1 = 2000 0 (adressendefinition)

2000 19          (fehlender index wird durch 0 ergaenzt)

n 1 90           (symbolische adresse n)

n 1 92

n+1 1 96         (symbolische adresse n+1, adressummand +1)

null 31          (symbolische adresse null durch bezugspunkt
                 definiert, wird in relative adresse uebersetzt)

0 1 91

pos: x1 1 35

2 1 93

n+1 1 97

pos 912          (symb. adre. pos durch bezugspunkt definiert, wird
                 in relative adresse uebersetzt)

s 32             (symb. adre., durch obige adresserdefinition er-
                 klaert, wird in absolute adresse uebersetzt)

9000 19

null:+0          (eine zahl wird in 14ziffrige gk-form uebersetzt
                 und auf ihrer programmposition wieder abgesetzt.
                 eine zahl kann nicht mit einer ziffer beginnen,
                 syntax 2.9.)
9999999

9999999


ausgabe nach fehlerstop


0009 9: iop mit adresse unvereinbar
```

```
(beispiel 10. symbol-programm fuer er 56 zur berechnung der
              funktion y = sqrt(abs(sin(x))).
              betriebsart: gleitkomma.
              daten: x = (0000,0001).
              bp: bp-sin(x) ab zelle 3000;
                  bp-sqrt(x) ab zelle 1000.
              ergebnis: y = (0000,0001).)

x = 0000 0  y= x 0  sin=3000 0sqrt =sin=
2000 0

2000 19  x 31  sin 10 (sprung in bp-sin(x))  300019(betrag von
                                                  sinus)
sqrt 10 (sprung in bp-sqrt(x))  y 32 (ergebnis abspeichern)

9000 19

9999999
9999999
```

fernschreibprotokoll der uebersetzung und des namenverzeichnisses

```
2000019
0000031
3000010
3000019
1000010
0000032
9000019
9999999
9999999
```

namenverzeichnis:

```
0000 0 = x
0000 0 = y
3000 0 = sin
1000 0 = sqrt
```

```
(beispiel 11. symbol-programm fuer er 56. aufgabe siehe beispiel
              10. lokalisiere den fehler.)

x = 0000 0  x= y 0  sin=3000 0sqrt =sin=
2000 0

2000 19  x 31  sin 10 (sprung in bp-sin(x))  300019(betrag von
                                                  sinus)
sqrt 10 (sprung in bp-sqrt(x))  y 32 (ergebnis abspeichern)

9000 19

99999999
99999999
```

ausgabe nach fehlerstop

0000 9: doppelte definition

(beispiel 12. symbol-programm fuer er 56. aufgabe siehe beispiel 2. zusatzforderung: nach programmstop soll das programm durch druecken der starttaste wieder gestartet werden koennen.)

```
      s, n, x1 = 2000 0
stop: 9000 19
      2000 19   (start)
       n 1 90
       n 1 92
     n+1 1 96
      null 31
       0 1 91
pos:  x1 1 35
       2 1 93
     n+1 1 97
      pos  12
       s   32
      stop 10   (unbedingter sprung zum programmanfang)
null: +0
      9999999
      9999999
```

fernschreibprotokoll der uebersetzung und des namenverzeichnisses

```
9000019
2000019
2002190
2002192
2003196
0007931
0000191
2004135
0002193
2003197
9996912
2000032
9987910
1000000
0000000
9999999
9999999
```

namenverzeichnis:

```
2000 0 = s
2002 0 = n
2004 0 = x1
0000 9 = stop
0013 9 = null
0007 9 = pos
```

(beispiel 13. symbol-programm fuer er 56. aufgabe siehe beispiel 12, andere loesung. lokalisiere den fehler.)

```
        s, n, x1 = 2000 0
stop: 9000 19
      2000 19    (start)
       n 1 90
       n 1 92
     n+1 1 96
      null 31
       0 1 91
pos:  x1 1 35
       2 1 93
     n+1 1 97
      pos  12
       s   32
    stop 9 91
null: +0
      9999999
      9999999
```

ausgabe nach fehlerstop

0012 9: iop mit adresse unvereinbar

(beispiel 14. symbol-programm fuer er 56. aufgabe siehe
beispiel 12, andere loesung.)

```
        s, n, x1 = 2000 0
stop:  9000 19
       2000 19    (start)
        n 1 90
        n 1 92
      n+1 1 96
       null 31
        0 1 91
pos:    x1 1 35
        2 1 93
      n+1 1 97
       pos  12
        s   32
     stop 9 93
null:  +0
       9999999
       9999999
```

fernschreibprotokoll der uebersetzung und des namenverzeichnisses

```
9000019
2000019
2002190
2002192
2003196
0007931
0000191
2004135
0002193
2003197
9996912
2000032
9987993
1000000
0000000
9999999
9999999
```

namenverzeichnis:

```
2000 0 = s
2002 0 = n
2004 0 = x1
0000 9 = stop
0013 9 = null
0007 9 = pos
```

(beispiel 15. symbol-programm fuer er 56. aufgabe siehe
beispiel 12, andere loesung.)

```
      s, n, x1 = 2000 0
stop: 9000 19
      2000 19   (start)
       n 1 90
       n 1 92
     n+1 1 96
      null 31
       0 1 91
pos:  x1 1 35
       2 1 93
     n+1 1 97
      pos  12
       s   32
    stop 9 95
null: +0
      9999999
      9999999
```

fernschreibprotokoll der uebersetzung und des namenverzeichnisses

```
9000019
2000019
2002190
2002192
2003196
0007931
0000191
2004135
0002193
2003197
9996912
2000032
0013995
1000000
0000000
9999999
9999999
```

namenverzeichnis:

```
2000 0 = s
2002 0 = n
2004 0 = x1
0000 9 = stop
0013 9 = null
0007 9 = pos
```

```
(beispiel 16. symbol-programm fuer er 56. aufgabe siehe
              beispiel 2.
              modifikation: das summationsergebnis s soll vom
                            programm in der form
                                 'ergebnis
                                  s = (zahlenwert)'
                            ausgeschrieben werden. abspeichern
                            des ergebnisses nach zelle (2000,
                            2001) kann unterbleiben.)

      n,x1 = 2002 0
      2000 19  n 1 90  n 1 92  n+1 1 96  nul  31  0 1 91
pos:  x1 1 35  2 1 93  n+1 1 97  pos 12  s 30  text 69  9000 19
text: '
                    ergebnis

                    s = '

null: s: +0
9999999 9999999
```

fernschreibprotokoll der uebersetzung und des namenverzeichnisses

```
2000019
2002190
2002192
2003196
0028931
0000191
2004135
0002193
2003197
9996912
0022932
0001969
9000019
3424139
3393939
3393939
3393939
3393939
3393939
3393939
3393965
3796765
3627469
3804241
3424139
3393939
3393939
3393939
3393939
3393939
3393939
3393980
6391039
1000000
0000000
9999999
9999999
```

namenverzeichnis:

```
2002 0 = n
2004 0 = x1
0033 9 = null
0006 9 = pos
0033 9 = s
0013 9 = text
```

(beispiel 17. symbol-programm fuer er 56. aufgabe siehe beispiel 12, andere loesung. lokalisiere den programmfehler. wie er behoben werden kann siehe syntax 3.2..)

```
      s, n = 2000 0
stop: 9000 19
      2000 19   (start)
       n 1 90
       n 1 92
     n+1 1 96
      null 31
       0 1 91
pos:  x1 1 35
       2 1 93
     n+1 1 97
      pos  12
       s   32
    stop 9 93
null: +0
      9999999
      9999999
```

ausgabe nach fehlerstop

x1 nicht definiert

ausgabe nach erneutem start, wobei ein lochstreifen mit der information

```
x1 = 2004 0
9999999 9999999
```

im fotoleser bereitliegt.

```
9000019
2000019
2002190
2002192
2003196
0007931
0000191
2004135
0002193
2003197
9996912
2000032
9987993
1000000
0000000
9999999
9999999
```

namenverzeichnis:

```
2000 0 = s
2002 0 = n
0000 9 = stop
0013 9 = null
0007 9 = pos
2004 0 = x1
```