Wigton Museum
www.t-lcarchive.org

22-01-2020

EL-X1 ALGOL 60
Compiler Notebook

t-lcarchive.org

E. W. Dijkstra

De gedetailleerde gang van zaken voor de eerstkomende dagen stel ik me als volgt voor. Morgenochtend, terwijl de handcode-band up to date gemaakt wordt, kunnen we al onze testgevallen in duplo op een lange band reproduceren. Het eerste exemplaar dient dan afgesloten te worden door een stukje blank tape, het tweede exemplaar dient na vier pentades blank een pentade = 4 te hebben. Dit moet straks aan de X1 aanmerkelijk handiger zijn, dan al die losse bandjes; tevens zijn we de losse doos-jes kwijt.

Aan de machine verrichten we de volgende handelingen. We beginnen met het bandje van Dekker, dat het geheugen vol met 7P-opdrachten zet. Daarna lezen we de voorponsing 2CP in, dan de nieuwe handcodeband en daaroverheen de oude versie der pagina's ZF0 en ZF1; daarna de band 2CP0-1, maar GVC0 moet 128 zijn

Dan leggen we band met testgevallen in de bandlezer, maken het consolewoord negatief en geven, na gecontroleerd te hebben, dat de ponser aanstaat, autostart 0: de machine stopt na compilatie van de eerste test. Aldus doen we voor alle testgevallen, iedere keer de X1 via autostart 0 startend.

De band, die hierbij geproduceerd wordt, vergelijken we met de oude resultaatbandjes: ik neem aan, dat we overeenstem-ming zullen constateren, met uitzondering van de extra uitge-ponste GVC0.

We lezen dan opnieuw de voorponsing in, dan de nieuwe versie van de kanalen ZF0 en ZF1 en maken dan onmiddelijk een biband. Ik neem nl. aan dat de controle, dat de nieuwe ponsroutine correct is, vrij veel tijd in beslag zal nemen en het lijkt me daarom prettig, om maar vast op hoop van ze-gen een biband te hebben.

Eigenlijk stel ik me voor, om, na weer de band 2CP0-1 in te hebben gelezen (deze immers wordt door de voorponsing overschreven) meteen het hele set opnieuw te produceren. We moeten er nl. rekening mee houden, dat de nieuwe ponsroutine goed is. We kunnen ons dan van de machine terugtrekken en de resultaatband op ons gemak bestuderen.

Ik denk niet, dat het nodig is, om van de resultaatband dan alles te controleren, het lijkt me nl. een ontzettend werk. Wellicht kunnen we iemand anders aantrekken, om deze

tkarchive.org

band te ontcijferen; beter nog kunnen we dit over een paar mensen verdelen. Totdat we de uitslag van deze controle hebben kunnen we niet veel meer doen. Wel moeten we een copie van de band 2CPO-1 hebben: een exemplaar wordt bij de biband opgeborgen, een exemplaar bij de voorponsing 2CP.

Onafhankelijk hiervan kunnen we nog wat testgevallen maken. Hier moeten bij zijn alle speciale functies, incl. "read" en "print", omdat dit de enige manier is, om te controleren, dat we ons met de prevulling van NLI niet hebben vergist. Verder is het noodzakelijk, een testgeval te construeren, waarbij de identifier van een speciale functie als parameter wordt meegegeven, zowel een keer als loze identifier, als als expressie, bv. sin(x).

Als ik flink ben, ga ik eveneens achter het volgende geval aan: ik herinner me, dat we, bij de bepaling, hoe lang de X1 nu eigenlijk rekende, aanvankelijk alleen het ponsen hebben kortgesloten in FOB6. Toen dit niet werkte, hebben we eveneens het typen kortgesloten en toen ging het wel. Achteraf is het mij niet duidelijk, wat voor fout er toen gemaakt is, het zou volgens mij ook met enkel typen moeten werken. Het lijkt me verstandig, om hier achteraan te gaan: het zou kunnen zijn, dat FOB6 een logische fout herbergt, die nog niet aan het licht is getreden omdat bv. al onze testgevallen output limited zijn geweest.

thearchive.org

Om de vertaler in te lezen legt men de (nieuwste versie van de) MC-Vertaler in de bandlezer en geeft, met CW[26]= 0, autostart 1. Na een paar meter is een zelf-controlerend stuk in handcode gelezen en gecontroleerd en het invoerprogramma stopt op OS. Om de rest in te lezen drukt men weer op toets 1 van het handregister. Hierna moet de bandlezer omgezet worden op 7-gatsband.

Voordat men nu de vertaler start controleert men, dat schrijfmachine en ponser aangeschakeld zijn en dat de band-voorraand in de ponser voldoende is.

Het vertaalproces zal de ALGOL-tekst twee maal lezen; pas bij de tweede keer wordt het objectprogramma uitgeponst. Als we van de ALGOL-tekst een band in duplo hebben, dan kunnen we de vertaler starten met CW[26]= 1; zo nee, dan moeten we de vertaler starten met CW[26]= 0, tengevolge waarvan de vertaler na de eerste lezing stopt. Men moet dan de band terugleggen en de X1 met BVA doorstarten.

Men start de vertaler met autostart 0.

Tijdens de vertaling worden de plaatsen van labels en beginpunten van procedures in de volgorde, waarin zij in de tekst voorkomen, uitgetypt. Deze plaatsbepaling geschiedt in het 32-tallig stelsel t.o.v het nog niet bekende beginpunt van het programma. Als de vertaling voltooid is, hoort de vertaler nog -0 uit te typen; als er iets anders uitgetypt is, is er een fout gemaakt door de X1 of waren de beide exemplaren van de tekst niet gelijkwaardig.(Deze min nul is nl. het resultaat van een somcheck op de van de band gedestilleerde ALGOL-symbolen. De beide exemplaren van de band hoeven dus niet heptade voor heptade aan elkaar gelijk te zijn!)

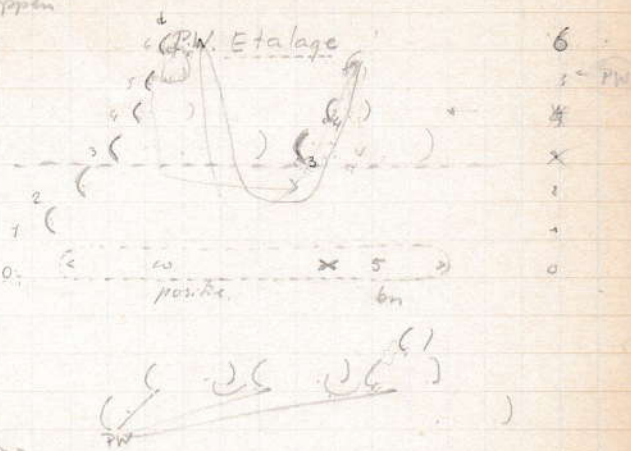
tiloerink.org

Om bij het aanroepen van een procedure altijd de koortsluiting er in te stoppen

toestands-grootheden LW ← nieuwe LW

eenvoudige PW ← nieuwe PW.

Toestandsgrootheden
LW, PW, NW, BN.



36	4P		SA	
60	3P	5	SS	positie in S
36	2LA	81	A	bn in A
36	4P		AB	
60	0S	0	ETA B	number BN[2n]
36	4P		SB	

288 µs.

Expliete aanroep: alleen nieuwe PW in etalage zetten; RETURN met PW & BN van nieuwe toestand eerzaak oplappen

Impl. subr. en formele procedure. PW en BN van block van call

FORMELE c implicatie: je begint met de combinatie PW & BN de etalage zijn te date te maken; voor de formele call roep je de subroutine dan normaal aan.

LW ← LW.
WW interessige v.d. procedure.
link
PW
BN ← maakt ik scherts.
← nieuwe PW

LW = 16 X1
WW = 17 X1
PW = 19 X1
BN = 20 X1

rt-archive.org

AL1 22-3-'60

(0) UDD UP DATE DISPLAY

[19 X1] = PW vom bestemming
[20 X1] = BN vom bestemming

		DA	o	Z	Eo	DI	
⇒	0	2S	19		X1		
	1	2A	20		X1	Z.	
	2	N 2T	6		Eo	A	→
	3	2T	8		Xo		⇒
9 ⇒	4	4P			SB		
	5	2S	0		Xo	B	
2 →	6	1A	1			A	Z
	7	4P			AB		
	8	6S	1		ETA	B	
	9	N 2T	4	Z	Eo	A	→
	10	2T	8		Xo		⇒

als bestemming = hoofdprogramma
nieuwe PW in S-register

cyclusduur 280 μs.

als bestemming ≠ hoofdprogramma.

t-1carchive.org

16 XI Switchindex.
17 XI WW
18 XI AW
19 XI PW
20 XI BN

1) LW wijst naar reconstructiegrootheden, in de stapel met name een ouder LW.

2) WW is gedurende de uitvoering van een blok constant, en geeft de EVP in de stapel tussen de assignments.

3) De PW in 19XI telt een sluimerend bestaan, is vooral van belang bij binnenkomst van een blok, als het nieuwe boknummer nog niet bekend is. En elke routine begint zijn eigen boknummer te vermelden en dander controle daarvan de etalage in te vullen.

4) BN wijst in wezen aan de plaats in de Etalage, tot waar de PW's van interesse zijn; bij TRANSHARE moet ETA[BN] in de stapel gezet worden; of dit getal identiek is aan [19 XI] staat nog te bezien bij de impliciete subroutine

5) AW is een dynamische grootheid, die de EVP in de stapel bijhoudt; de arithmetische operaties worden in termen van AW geprogrammeerd, bij call wordt AW uitgelezen, bij Return gezet. Tussen de statements in AW=WW

t-lcarchive.org

[2B . . BN$_a$ ⊕ [B] := BN$_a$
  SCC

t-lcarchive.org

Het mechanisme van de aanroep:
Zonder accenten is oud, staan te verdwijnen.

stapelbeeld

(AW →)	ETA[BN' - 1]	0	← PW'
	WW	1	} link-gegevens.
	terugkeeradres	2	
	ETA[BN]	3	
	BN	4	

- - - - - - - - - - - ← ' ← kortsluiting.

primitief zo belang.

Het boven geschilderde mechanisme geldt voor non-formele aanroepen.

Bij form. en impl. aanroep beginnen we met begeleidende PW-BN-combinatie; deze combinatie vstuurt na het redden als boven (act/m 4) de operatie UDD.
Daarna kan de aanroep van een formele procedure normaal verlopen, met de door de redoperatie ~~gecreëerde~~ geaffirmeerde PW'. = LW+5
Nu wil ik liefst: voor de impliciete subr. BN en PW introduceren

Nadat we UDD gedaan hebben, hogen we BN 1 keer op en doen de ~~...~~ kortsluiting.

SCC(0) Short CirCuit met [B] = BN'

```
      DA  0  ZE0  D1
⇒ 0   6B  20  X1
  1   2S  19  X1        PW in S     ] nieuwe PW in ETALAGE
  2   6S  0 ETA B
  3   2A  1- ETA B                  ]
  4   4P      SB                    ] Kortsluiting in Stapel
  5   6A  0   X0 B                  ]
  6   2T  8   X0    E ⇒
```

420 μs.

SCC wordt aangeroepen aan het begin van en door elke procedure.
en verder aan het einde van FORM-TRANSMARK voor het geval we een impliciete subroutine zullen uitvoeren.
maakt nl. een schachtniveau 1 hoger =: BN'

AL5 ; 23-3-60.

ADRESLOOS		ADRESHEBBEND	TWEEDE OPERATOR
	+	+	Integer
	−	−	Floating
	×	×	Formeel.
	/	/	
	÷	÷	
	↑	↷	
	:−		
	:=		
	↷	↷#	Integer
			Floating
			FORMEEL.

Indexer Levert een adres in de jongste accumulator.

In een accumulator staat óf een adres óf een waarde; dit is er wel meestal te aanzien in aangegeven, maar dit weet het omringende programma;
Wel is in een accumulator aangegeven, of het gaat om drijvend dan wel vast getal.
De adresloze operaties zijn typeloos in hun formulering; bij de uitvoering kijken ze naar de beide accumulatoren! Hiervoor zijn vaste regels.
drijvend is preferent.
Overloop bij integers ~~geeft~~ geeft overgang naar floating.
(Uit detectie overwegingen moet dit op een aanwijsbaar punt!)

a[i,j]. → ↷# a
            ↷ i
            ↷ j

Levert me steeg functie en type.

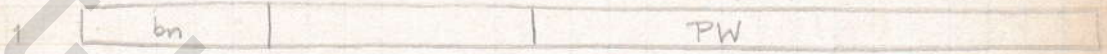
X := a + b * c

↷# X
↷ a
↷ b F
* c
+
:=

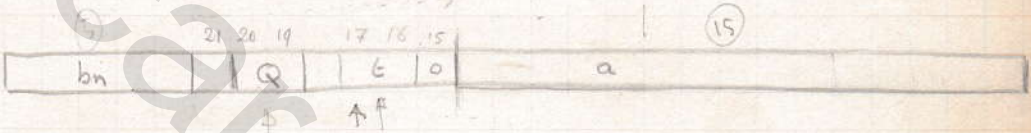
t-lacchive.org

Structuur PARD

Q = 0 , a = adres floating Q = 2 , a = beginadres subr met numeriek antwoord
= 1 , a = adres integer. Q = 3 : a = beginadres subr met adresantwoord.



Structuur PORD



t = 0 r = a
t = 1 r = PW [bn] + a
t = 3 name ongeïnterpreteerd transporteren (PW[bn] + a) & (PW [bn] + a + 1)

Vermenging stapelbeeld

AW → RW
ETA[BN-1] ← PW
WW
terug leesadres
ETA [BN]
BN

(19X1)

RW = 0 als geen resultaat gewenst (codenom FAIL)
RW > 0 , n.l. RW = physisch adres van het resultaat.

voor t = even kiezen we in TLI een codering voor de hulpmiddelen d_{25} d_{24} :

00	oXo
01	RLI
10	FLI
11	KLI

zie 6a.

tl-archive.org

17 WN α = accumulatorlenght (usch. 4)
18 AW *
19 PW
20 BN

0 AW → RW ✓ *
1 ETH[BN'-1] 0 ← PW *
2 WN 1 v
3 Tempspeicher 2 ✓
4 (19 X/) 3 v
5 BN 4 v
6 5

AL6a 21-4-60.

REPRESENTATIE VAN PORD'S in TLI

Voor t = oneven: de PORD zelve.

Voor t = even een codering voor het nulpunt, nl.

d_{25} d_{24} :	00	voor	X0
	01	voor	RLI
	10	voor	FLI
	11	voor	KLI

t-lcarchive.org

EXTRANSMARK (1) ETM R = Resultaat P = Procedure AL 7 24-3-'60

[A] = aantal PORDS
[B] = bestemming = begin adres

DPZF "FRM"

		DA	o	Z	Eo	Di	
ETMR ⇒	0	3S	4		A		ophoging
	1	5S	18	X1			AW
	2	0S	18	X1			RW = 0
	3	2T	5	Eo	A	⇒	
ETMP ⇒	4	3S	0		A		RW = -o
3 →	5	6A	0	Xo			berg. aantal PORDS
	6	6B	2	X1			2X1 = +o „dn. niet perform.
	7	6B	3	X1			bestemming
	8	2A	9	Xo			pak link
	9	2B	18	X1			AW
→	10	6S	0	Xo	B		RW
	11	6A	3	Xo	B		terugkeeradres
	12	2A	17	X1			
	13	6A	2	Xo	B		bW
	14	2A	19	X1			
	15	6A	4	Xo	B		PW
	16	2A	20	X1			
	17	6A	5	Xo	B		BN
	18	1P	5	AA			
	19	0A	19	X1			+ PW
	20	6A	4	X1			2de PORDWOORD
	21	0B	1		A		
	22	6B	19	X1			nieuwe PW
	23	0B	5		A		
	24	0B	0	Xo			zet vooral
	25	0B	0	Xo			AW
	26	6B	17	X1			bW
	27	6B	18	X1			
	28	2T	26	E1	A	⇒	
26 E1 ⇒	29	2B	9	Xo			
	30	1B	0	Xo			
	31	2A	32765	Xo	B		pak PORD

		DA	o	Z	E1	Di	
	0	U	2LA	30	E1	Z	d16
	1	Y	2T	14	E1	A	→ t = 0 of 2
	2		6A	5	X1		red PORD
	3		0P	5	SA		
	4		2LS	31		A	bn in S
	5		4P		SB		in B
	6		2B	o	ETA	B	PW[bn] = S
	7		0B	5	X1		verwijs
	8		2A	5	X1		
	9	U	2LA	31	Z E1	Z	d17
	10	Y	2T	17	E1	A	→ t = 1
	11		2A	o	Xo	B	t = 3 transport
	12		2S	1	Xo	B	zuiver formele
	13		2T	21	E1	A	⇒ identificeer
1 ⇒	14	U	2LA	31	E1	Z	d17
	15		4P		AB		
	16	N	2B	o	Xo	B	neem in bedrijf van
10 →	17		2LA				3 x 2^19
	18		6B	5	X1		tel adres
	19		0A	5	X1		bij Q_2^19
	20		2S	4	X1		2de PORDWOORD
13 →	21		2B	19	X1		
	22		0B	o	Xo		berg
	23		0B	o	Xo		PORD
	24		6A	5	Xo	B	
	25		6S	6	Xo	B	
28 Eo →	26		4T	29	Eo	o E	→ volgende PORD
	27		2A	2	X1	P	
	28	N	2T	8	F1	A	→ berg in FRM
	29		2T	3	X1		⇒ naar PROCEDURE
	30	Y	0A	o	Xo		d16
	31		0A	o	Xo	P	d17

voor n PORD's: tijdduur $(n+1) * 1{,}5$ ms.

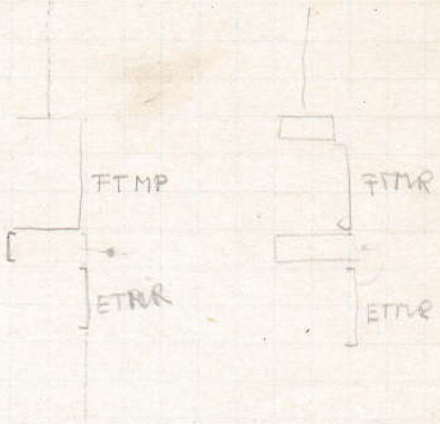
t-tarchive.org

, JOHAN (x, y, z), 'STORE OF GIVEN IDENTIFIER

(UP,)

, UP, адрес FTMP

X := UP



t-1carchive.org

AL8

[FTMR FORMTRANSMARK RESULT
FTMP FORMTRANSMARK PROCEDURE] (eventueel met parameter) 20-4-'60
TFR TAKE FORMAL RESULT
TFA TAKE FORMAL ADRESS] PARD-positie in [S]

FRM

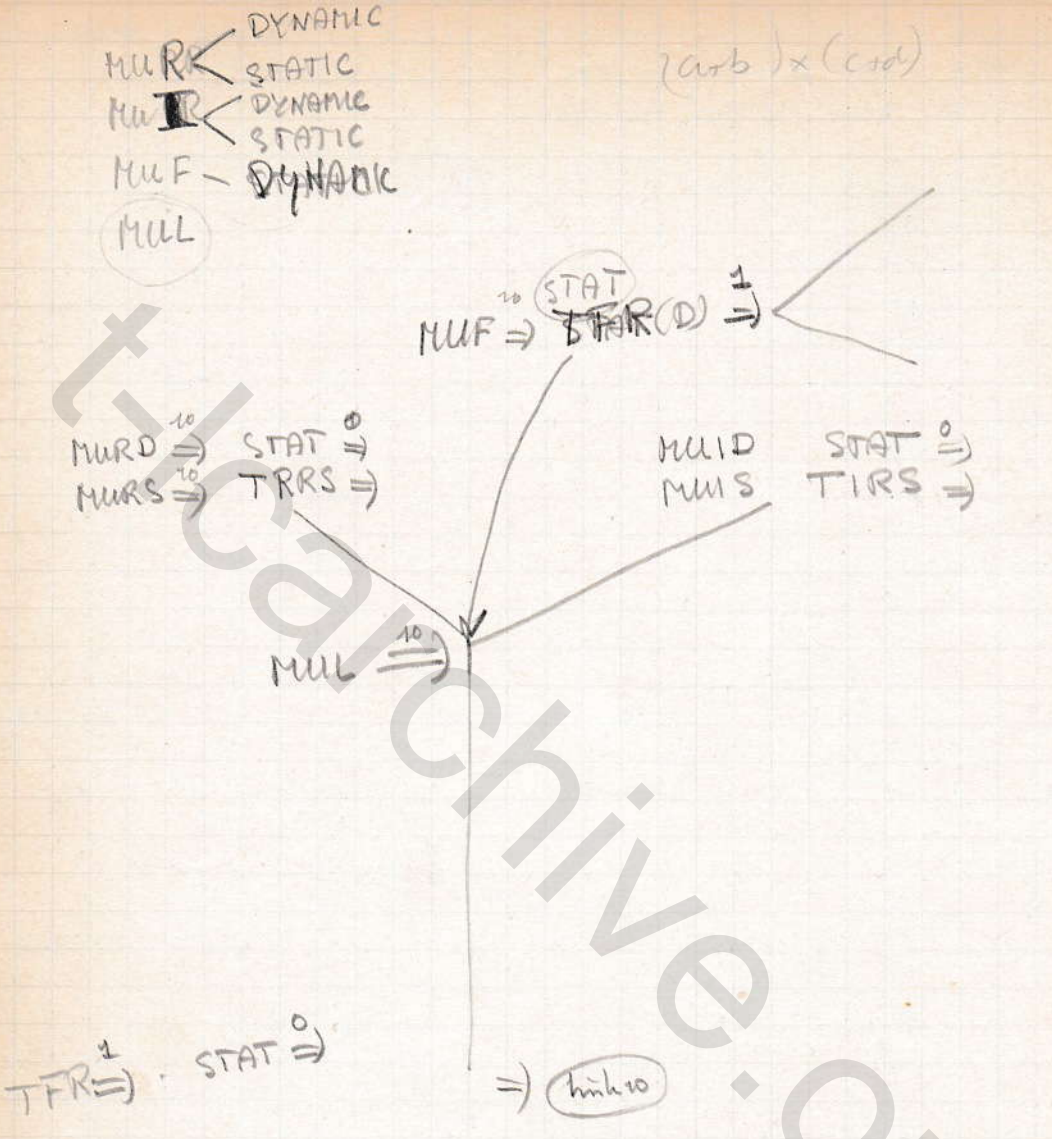
in [A] = aantal parameters.

```
      DPZL "TAR"
      DPZK "SCC"
      DPZH "UDD"
      DPZF "ETM"
```

```
                DA o Z Eo DI
TFA ⇒   0       2B 1     A
        1       2A 0     A
        2       2T 5     Eo A  ⇒
TFR ⇒   3       2A 0     A
FTMR ⇒  4       2B 0     A
  2 →   5       6A 0     Xo        aantal PARD'S
        6       3A 4     A         ophoging
        7       5A 18    X1        AW
        8       oA 18    X1        RW in A
        9       2T 13    Eo A  ⇒
FTMP    10      6A 0     Xo
        11      3A 0     A         negatieve RW
        12      2B 1     A
  9 →   13      7B 2     X1        =1 : met adres levreden
        14      6A 3     X1
        15      6T "STAT" 0  ⇒
        16      2S 0     Xo B
        17    U 2LS            Z   d20=0?
        18    N 2T 25    Eo A      → schrik van leven
        19      3A 2     X1  Z     willekeurig getal
        20    Y 2T                → naar TAR1 mdq=[S]
        21      2B 3     X1  P
        22    Y 6S 0     Xo B      berg adres
        23    Y 6A 3     Xo B      zet typevlag = +1
        24      2T 9     Xo  E ⇒
  18 ⇒  25      3A 2     X1  Z
        26    N 3LS                met dig
        27      2A 1     Xo B      staart PARD
        28      2B 18    X1
        29      6S 2     Xo B      kop van PARD
        30      2S 9     Xo
        31      6S 0     Xo B
```

```
                DA o Z E1 DI
        0       2S 10    Xo
        1       6S 1     Xo B
        2       6A 1     X1
        3       2S 3     X1
        4       2B 3       A
        5       oB 18    X1
        6       2A 19    E1 A
        7       2T 10 Z  Fo A  ⇒
ETM ⇒   8       2S 1     X1      2de PARDWOORD
        9       4P       SA
        10      2LS 32767  A
        11      oP 5     AA
        12      2LA 31     A Z
        13      6A 20    X1
        14    N 6T 6     Ho o    ⇒ UDD
        15      2B 1       A
        16      oB 20    X1
        17      6T 0     Ko o    ⇒ SCC
        18      2T 32766 Xo B    ⇒ noarsubr.
RETURN ⇒ 19     3B 3       A
        20      oB 18    X1
        21      6B 18    X1
        22      2A 0     Xo B
        23      6A 9     Xo
        24      2A 1     Xo B
        25      6A 10    Xo
        26      2A 2     Xo B
        27      2LA                Z dig
        28    Y 2T 9     Xo      E →
        29      2T o     Lo A    →
        30
        31
```

t-iarchive.org



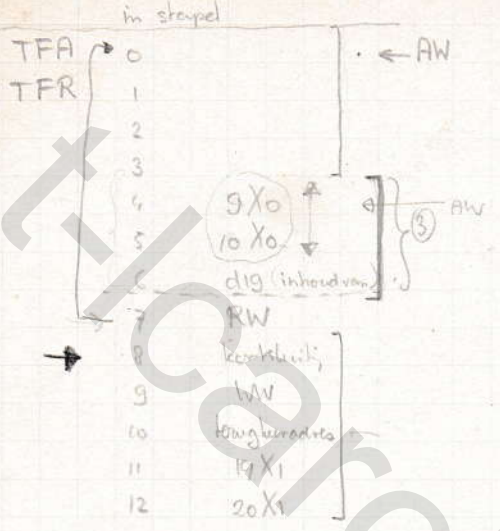
tricarchive.org

FTMP.
TFR
TFA

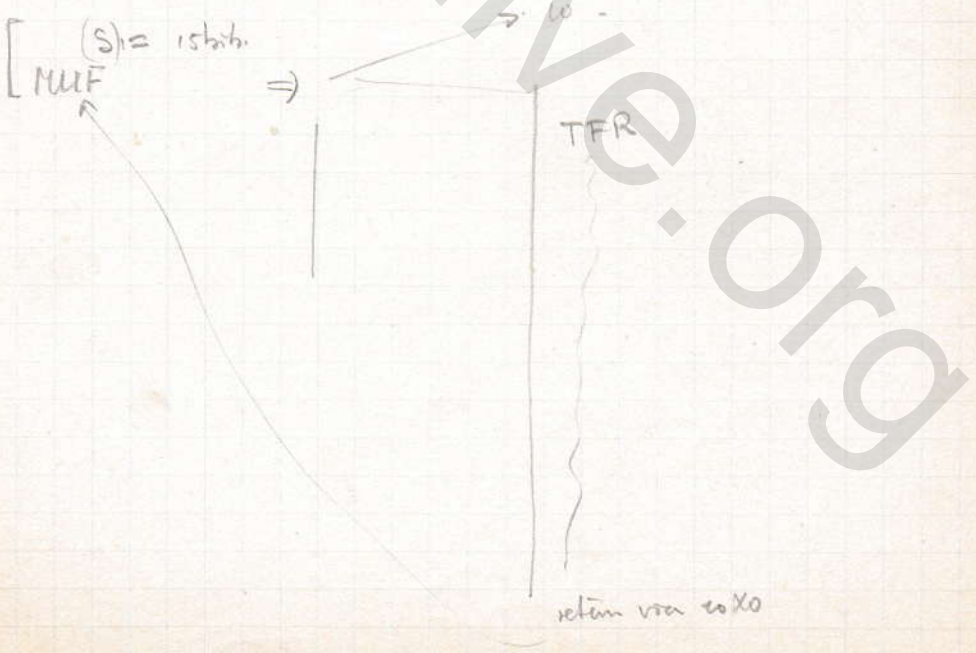
AL10 20.4.'60

ALG in orde wet

FTMP



0	9 X0	← AW
1	10 X0	
2	dig	(speel als "resultaat acceptabel")
3	RW (≤0)	
4	korkslurk	← PW
5	WW	
6	terugkeer	
7	19 X1	
8	20 X1	



tlcarchive.org

FORMTRANSMARK(0) FTM R resultaat
P procedure. 063 AL8 24-3-'60

~~DP2K "TAK"~~
DP2H "UDD"
DP2F "ETM"

[A]= aantal PARD=
[B]= PARD. POSITIE.

DA 0 Z E0 DI

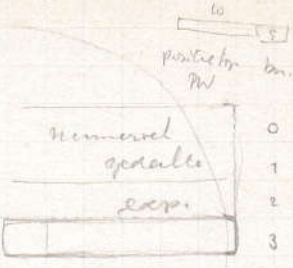
tijdsduur (n+2) * 1,5 ms

```
FTMR⇒0    3S 4      A      opkomen
          1 5S 18   XI     van AW
          2 0S 18   XI     RW≥0
          3 2T 5    E0 A ⇒
FTMF⇒4    3S 0      A      RW<0.
3→        5 6A 0    X0     aantal
          6 4P      BA
          7 3P 5    AA     maak vam
          8 6A 0    XI     PARD-
          9 4P      BA     POSITIE
         10 2LA 31      A   een
         11 4P      AB     PARD
         12 2B 0 ETA B     adres
         13 0B 0   XI
         14 2A 1   X0 B
         15 6A 1   XI     2de PARDWARD.
         16 2B 0   X0 B   adres uit de PARD.
         17 7B 2   XI     2XI≤-0.
         18 2T 7 Z F0 A ⇒ naar ETM
         19 0A 0   X0 C
uit
ETM →    20 2S 1   XI     2de PARDWORD
         21 4P      SA
         22 2LS 32767  A  in S.PW
         23 0P 5   AA
         24 2LA 31     A Z  in A.BN
         25 N 6T 6 H0 0 ⇒ UDD.
         26 2T 3   XI   ⇒ naar subr.
```

rijcarchive.org

(1) TFR Take Formal Result
(1) TFA Take Formal Address

[S] = PARD-Positie



3 positie (adres à 18)

(3) ≤ 0 Integer Filled.

```
                DPZK "TAR"
                DPZH " SCC "
                DPZF "UDD"
                   DA 0  Z EO DI
TFA ⇒ 0            2A 2    EO      dig
      1            2T 4    EO A    ⇒
      2            0A 0    XO A    | dig
TFR ⇒ 3            2A 0       A
  1 → 4            6A 2    X1
      5            4P      SA
      6            3P 5    SS      S = positie
      7           24A 31       A   bn
      8            4P         AB   bn = B
      9            0S 0 ETA B      in S adres van PARD
     10            4P         SB
     11            2S 0    XO B    kop van PARD
     12         U 2LS                 Z  d20
     13         N 2T 21    EO A    → schrik van oneindig
     14           2A 2     X1      Z  willen Resultaat?
     15         Y 2T                → naar TAR1 met q = [S]
     16           2B 18    X1
     17           6S 3     XO B
     18           0B 4        A
     19           6B 18    X1
     20           2T 9     XO      E ⇒
  13 ⇒ 21         8LS 2    X1
     22           2A 1     XO B    staat van PARD
     23           2B 18    X1
     24           6S 6     XO B    kop PARD
     25           2S 9     XO
     26           6S 4     XO B
     27           2S 10    XO
     28           6S 5     XO B
     29           6B 7     XO B
     30           2S 17    X1
     31           6S 9     XO B
```

```
 0 (                    ← AW
 1
 2                      acc
 3
 4      9 XO
 5      10 XO
 6   kop PARD dus dig'
 7      RW          →
 8   karakteristiek.   ← PW
 9      WN
10   lees bovenaan
11      19 X1
12      BN
```

t-larchive.org

Acc.

0	adres + 1B(=2^{19})	mlège	numerieke gedeelte.
1			
2			exponent
3	≠+0 als inhoud adres;	=-0 als integer	=-1 als drijvend.

t-lcarchive.org

Obs. AL 10 25-3-'60

Vervolg TFR, TFA

```
     DA  O  Z  E1  D1
0    2S  19    X1
1    6S  11    Xo  B   } PW
2    2S  20    X1
3    6S  12    Xo  B   } BN
4    0B  8       A
5    6B  19    X1        nieuwe PW.
6    0B  5       A
7    6B  17    X1        nieuwe HW
8    6B  18    X1        nieuwe AW
9    4P        AS
10   2L  32767   A       isoleer PW
11   0P  5       AA      BN rond
12   2LA 31      A  Z
13   6A  20    X1        nieuwe BN
14 N 6T  6     Fo  o   ⇒ UDD
15   2B  20    X1
16   0B  1       A
17   6T  0     Ho  o   ⇒ SCC  terug met PW in de B-lijn
18   2A  21    E1  A
19   6A  2     Xo  B
20   2T  32766 Xo  B   ⇒ naar begin subroutine
     Begin van Subroutine
⇒ 21 2B  18    X1
22   1B  3       A
23   6B  18    X1
24   2A  0     Xo  B
25   6A  9     Xo
26   2A  1     Xo  B
27   6A  10    Xo
28   2A  2     Xo  B
29   2LA 2     Eo  Z
30 Y 2T  9     Xo  E   → klaar.
31   2T  0     KO  A   ⇒ naar TAR (adreslooz).
```

ti-archive.org

RETURN **RET**. AL II 29-3-'60

DPZF "UDD"

	DA	o	Z	Eo	DI	
→ 0	2B	19		X1		
1	1B	1		A		
2	6B	18		X1		nieuwe AW
3	2S	2		Xo	B	] WW
4	6S	17		X1		
5	2S	3		Xo	B	
6	6S	8		Xo		
7	2S	4		Xo	B	
8	6S	19		X1		
9	2A	5		Xo	B Z	
10	6A	20		X1		
11	2T	2		Fo	A	⇒ via UDD terug

GTA (0) **Goto adjustment** [B] = bn van bestemmjo tolele.

λo ⇒ 0	6B	20		X1		
1	2S	o		ETA	B	
2	2B	19		X1		
→ 3	U 1S	3		Xo	B Z	
4	N 2B	3		Xo	B	
5	N 2T	3		Eo	A	→
6	2A	1		Xo	B	
7	6A	17		X1		WW
8	6A	18		X1		AW
9	6S	19		X1		
10	2T	8		Xo		⇒

t-1archive.org

- ID karakterisering van laatst gelezen identifier, in de vorm van verwijzing naar de naamlijst, of constante

NFLA = Name flag: =1 → ID nog actueel

AFLA = Address Flag =1 → jongste accumulator bevat adres,
=0 → jongste, indien aanwezig, bevat waarde.

OFLA = Operator flag =1 → + teken verwerpelijk
=0 → + teken betekent echte additie.

IFLA Index flag =1 → index situatie
=0 → non.

EFLA Expressie vlag =1 → Expressie - situatie; behandeld ":=" := ":-"
=0 →

VFLA For FLAg =1 dan in for clause - situatie

JFLA JumpFLAg =1 dan designational identifier

- IDL next delimiter

PFLA Procedure Flag =1 ID = procedure identifier

RFLA aRray Flag =1 als bezig met array-declatie

FFLA Formal Flag =1 ID is formeel

MFLA paraMeter Flag =1 Parameter situatie

BN = Bloknummer

FORA } exclusieve werkwoorden voor FOR-translation.
FORC }

PSTA } exclusieve werkw. Procedure Statement. ; bij komma 3.
PSTB }

rtlarchive.org

Operator hoogte. OH1

0	begin	([
1	;	]) end ,
2	:=	
3	≡	
4	⊂	
5	∨	
6	∧	
7	¬	
8	> ≥ = ≤ < ≠	
9	+ −	
10	* / ÷	NEG.
11	↑	
12	abs, sign, sqrt, sin, cos, arctan, ln, exp,	

t-1archive.org

(((switchdeclaratie

FLSC voor overspringen.
ID van het switch identificeer
--> :=
--> RLSC
switchkomma.

t-lcarchive.org

DDEL":="

```
        QFLA := 1;
        if VFLA = 1 then goto WORDT 1;
        if SFLA = 1 then goto WORDT2;
        if EFLA = 1 then DL := ":-" else EFLA := 1;
        OH1 := 2;
        if PFLA = 1 then
begin   FTL(BN(ID)); DL := DL + "!" end else
        if NFLA = 1 then GA1;
        FTD; goto EL;
-------------------------------------------------
WORDT1: EFLA := 1
        if NFLA = 1 then GA1;
        FRL("6T "FOR1" ? ");
        FRL("2T DECIM(FLSC) FL1 ");
        FORC := FLSC; FLSC := FLSC + 1;
        FL1[FORA] := RLSC;
        FRL("2A 0A");
        FRL("2B DECIM(FLSC) FL1 ");
        FORA := FLSC; FLSC := FLSC + 1;
        FRL(" 6T "ETMP" 0 "); goto EL;
-------------------------------------------------
WORDT2: FRL(" 2T DECIM(FLSC) FL1 ")
        FTL(FLSC); FLSC := FLSC + 1;
        FTL(ID)
        OH1 := 0
        FTD;
WORDT3: DL := 160
        FTL(RLSC)
        FTD;
        goto EL
```

tiarchive.org

AL 15 30-3-60

→ GAI Productie objectprogramma: Genereer adres van Identifier ID in volgende accumulator.

FTL(x) : Voeg x toe aan TLI

FTD : Voeg DL & OHI toe aan TLI

BN(X) : bepaal het blokenummer, dat behoort bij de verwijzing x naar de naamlijst.

POP : Productie Objectprogramma, afh. van OHI en stoppers.

LTF(x) Inverse van FTL

Obj SSI : STORE SWITCH INDEX.

FRL(x) : voeg X toe aan RLI

RNS : Read Next Symbol → DL
ingelijfd bij POP.

EVA : EVALUATE: vervang zonodig jongste accu door zijn inhoud

→ GRI Productie objectprogramma: Genereer Result van Identifier ID in volgende accumulator.

Obc. CAC Copy boolean Accumulator into Condition.

Obj. EIS End of Implicit Subroutine: voorlopig resultaat Toekenning aan eigen identifier vóór "return"

BPR Begin Procedure in Register. nl onder controle van ID bij expliciete subroutine physisch adres in B; bij formele procedure PARD-positie in S;

TPORD : geeft aan zich zelf in TLI-conventie een PORD afgeleid uit ID als lege Identifier.

BOID : Produceer sprong-opdracht adres van ID. (2T - RIA of 2T .... FLI).
AVR : Adres Variabele in Register. dus. statisch in B of dynamisch in S

t-lcarchive.org

DDEL "["

AL16 30-3-66

```
OFLA := 1;

if RFLA=1 then goto ..... ;        (ARRAY declaratie)
OHI:=0
FTL (EFLA); EFLA := 1;
FTL (IFLA); IFLA := 1;
FTL (MFLA); MFLA := 0;
FTL (FFLA);
FTL (JFLA);
FTL (ID);

FTD; if JFLA=0 then GAI; goto EL;
```

ttcarchive.org

```
DDEL ", "
    OFLA:=1
    if IFLA =1 then

begin

KOMMA1: OH1:=1; POP; if THENELSE then goto KOMMA1;

    OFLA:=1; goto EL
end;

    if VFLA =1 then
begin KOMMA2; OH1:=1; POP; if THENELSE then goto KOMMA2;

    if TLI[TLSC-1] = "for" then FRL("6T "FOR2"?") else

    begin TLSC:= TLSC-1;
          if TLI[TLSC]="while" then FRL("6T "FOR4"?") else
                                    FRL("6T "FOR7"?")

    end;
    goto EL
end;

    if MFLA=1 then goto KOMMA3;

    if SFLA=1 then goto KOMMA7;
```

tilarchive.org

(afsluiting van een parameter)

```
KOMMA3: if TLI[TLSC-1] ≠ "," then

begin KOMMA4; OHI:=1; POP; if THENELSE then goto KOMMA4;
   if TLI[TLSC-1] = "do" then
   begin TLSC := TLSC-3; BN:= BN-1;
      FRL("2T DECIM(TLI[TLSC]) RLI A");
      FLI[TLI[TLSC+1]] := RLSC; goto KOMMA4
   end;                                          } if DOT goto KOMMA4

   TLI[TLSC-2] := TLI[TLSC-2] + 1*2^24 + 2*2^19 + 0*2^16;
                  RLSC          RLI     Q=2      t=0
KOMMA5: FRL("2T "EIS" A")
end
else
begin if AFLA = 1 then
   begin TLI[TLSC-2] := TLI[TLSC-2] + 1*2^24 + 3*2^19 + 0*2^16; goto KOMMA5
   end;                               RLI     Q=3      t=0
                  RLSC

   if TLI[TLSC-2] ≠ RLSC then goto KOMMA4;

   if JFLA=1 ∧ FFLA=0 ∨ NFLA=0 then goto KOMMA4;

   TLI[TLSC-2] := TPORD
end;

   if DL = "," then goto OPEN2;

   PSTA := 0

KOMMA6: LTF(PSTB); if PSTB = "," then
begin  PSTA := PSTA + 1; LTF(PSTB);
   if PSTB ∧ d16 = 0 then
      FRL(PSTB ∧ (-3*2^24), "2^-24*(PSTB ∧ 3*2^24)") else
      FRL(PSTB, 0x0); goto KOMMA6
end
   TLSC := TLSC-1; FLI[TLI[TLSC]] := RLSC;
   FRL("2A DECIM(PSTA) A");
```

↓ voltooiing Procedure statement.
tellen e.d., transporteren van PORD

00 = 0X0
01 = RLI
10 = FLI
11 = KLI

tilarchive.org

```
BN := BN-1 ;
LTF(FFLA) ; LTF(EFLA);
if EFLA = 0 then
        begin if FFLA = 0 then FRL("6T  "ETMP" 1") else
                               FRL("6T "FTMP" 1")
        end
        else
        begin if FFLA = 0 then FRL("6T "ETMR" 1") else
                               FRL("6T "FTMR" 1")
        end;
AFLA := 0;
LTF(MFLA);
LTF(VFLA);
LTF(IFLA);
goto EL;
```

PTM

tt-archive.org

KOMMA 7: ALI7C 10-5-60

OHI:=1; POP; if THENELSE then goto KOMMA 7; OHI:=0; goto WORDS

t-1carchive.org

DDEL "]" AL18 30-3-60

```
if IFLA=0 then goto ;

BUS1: OHI:= 1 ; POP ; if THENELSE then goto BUS1 ;
      TLSC := TLSC - 1 ; comment verwyder [ ;

      LTF ( ID ) ;
      LTF (JFLA) ;
      LTF (FFLA) ;
      LTF (MFLA) ;
      LTF (IFLA) ;
      LTF (EFLA) ;

      if JFLA =1 then          Store Switch Index
                               ↑
begin NFLA := 1 ; FRL ("GT "SSI" 1 "); RNS ; goto DDEL end ;

      FRL ( GT "INDEXER" ? ); AFLA := 1 ; goto EL ;
```

tlarchive.org

DDEL "+ -" ; AL 19 31-3-'60

```
if RFLA = 1 then goto                    ; array declaratie.
if OFLA = 1 then

begin if DL = "+" then goto EL ;
OHI := 10 ; DL := NEG ; FTD ; goto EL end ;

      ; OHI := 9 ; POP ; FTD ;          ; goto EL ;
```

t-lcarchive.org

EVA EVALUATE AL20 31-3-60

```
begin if AFLA = 1 ∧ NFLA = 0 then
begin FRL ("6T "TAR" ?); AFLA := 0 end end;
```

t-lcarchive.org

AL 21 31-3-'60

DDEL " * / ÷ " ;

E : OH1 := 10; POP; FTD; goto EL;

t-lcarchive.org

AL 22 31-3-'60

```
DDEL "("
   OFLA := 1;
   if PFLA = 1 then goto OPEN1 ; procedure: functie

                  OH1 := 0;
   FTL (MFLA); MFLA := 0;
   FTD; goto EL;
```

```
OPEN1:  BPR; FRL("2T DECIM (FLSC) FL1 "); 19-4-'60
        FTL (IFLA); IFLA := 0;
        FTL (VFLA); VFLA := 0;
        FTL (MFLA); MFLA := 1;
        FTL (EFLA); EFLA := 1;
        FTL (FFLA);
        FTL (FLSC); FLSC := FLSC + 1;
        OH1 := 0; FTD; DL := "," ; BN := BN + 1;

OPEN2:  FTL (RLSC); OH1 := 0; FTD; AFLA := 0; goto EL;
```

FLSC
(
PORD
,
PORD
,

rijksarchief.org

```
DDEL ")"
    if MFLA = 1 then goto ;  { laatste parameter gehad.
    OHI := 1; POP; TLSC := TLSC - 1;
    LTF(MFLA);
    goto EL;
```

14-4-60.

```
DDEL")"
    if MFLA = 1 then goto KOMMA3;
DICHT1: OHI := 1; POP; if THENELSE then goto DICHT1;
    TLSC := TLSC - 1;
    LTF (MFLA); goto EL;
```

tlcarchive.org

AL24 31-3-'80

; x := (a - ((b + c) / d + e) * (- f + g)) * h ;

ID	IL	TLI (OHI)	RLI	FLA N	FLA A	FLA O
		begin 0				
x	:=	:= 2	∘ #x :	1	1	1
	(			0		1
		~~(0~~				
a	-	~~- 9~~	a :	1	0	0
	(			0		1
		~~(0~~				
	(			0		1
		~~(0~~				
b	+	~~+ 9~~	b :	1	0	0
c	)		+ c :	1		0
	/	~~/ 10~~		0		
d	+	~~+ 9~~	/ d :	1		0
e	)		+ e :	1		0
	*	~~* 10~~		0		0
	(	~~(0~~				1
	-	~~NEG 10~~				
f	+		f :	1	0	0
		~~+ 9~~	NEG .			
g	)		+ g :			
	)		* .	0	0	0
			- .			
	*	~~* 10~~		0	0	
h	;		* h :	1		
			:= .			

#x

~~#x~~

x := (a - ((b + c) / d + e) * (- f + g)) * h

- f + g

22 ops
27 symbols

M per FLA

tl-archive.org

x [i - j * k [h] , r * ℓ + s] := y ;

ID	DL	TLI	RLI	N A O I
		begin o		o
x	[	~~IFLA = 0~~	.	1 0 1 1
		~~[0~~	↷ # x :	0 1
i	-	~~- 9~~	↷ i :	1 1 0 1
j	*	~~* 10~~	↷ j :	1
k	[	~~IFLA+1~~	↷ # k :	0 1 1 1
		~~[0~~		
h	]		↷ h :	1 1 1 0
			INDEXER .	0 1 0 1
	,		↷ .	0
			* .	
			- .	
r	*	~~* 10~~	↷ r :	1 0 0 1
ℓ	+	~~+ 9~~	* ℓ .	
s	]		+ s :	
			Indexer	0 1 0 0
	:=	:= 2		0 1
y	;		↷ y :	1
			:= .	

(24)

x [ ] ~~i - j * k [h]~~ y
~~j * k [h]~~ r x ℓ + s
~~# k [h]~~

20 symb
24 woorden

20 symb → 24 woorden.

, over ,

a + b
- b

, a [i] ,

, a + b ,

, a ,

t-larchive.org

Procedure POP wat betreft arithmetiek en indicering?

```
begin if NFLA = 0 then
    begin if AFLA = 1 then
        begin AFLA:=0 ; FRL("GT "TAR" ?") end
    end NFLA=0 else

    begin if OHI ≤ OHIT[TLSC-1] then
        begin if OPA(TLSC-1) then
        begin TLSC:= TLSC-1 ;
            FRL ("OPT[TLSC] . ID")
        end
        end else GRI ; NFLA:=0
    end ;

POP1: if OHI ≤ OHIT[TLSC-1] then
        begin TLSC:= TLSC-1 ;
            FRL ("OPT[TLSC]");
            goto POP1
        end
end ;
```

Niet regelen op parameterloze functieprocedure!

en het regelen op de jump flag!

basisschema

TUbarchive.org

afkorting parameter

NFLA = 1 → TLI eindigt op , ~~of (~~ lore identifier.
NFLA = 1 → Expressie., niet

NFLA = 0 → TLI eindigt op , AFLA = 1 : gewijzigde nonexpressie.
NFLA = 0 → TLI eindigt op , AFLA = 0 ; expressie.
NFLA = 0 → TLI niet op , expressie.

TLI eindigt niet op , expressie en dus POP
of NFLA = 0 ∧ AFLA = 0 . " "

TLI eindigt op , NFLA = 1 lore identifier
= 0 gewijzigde nonexpressie.

t-larchive.org

EFLA
FLSC
then

EFLA
if

t-lcarchive.org

```
boolean procedure THENELSE ;
    begin if TLI[TLSC-1] = "then" v TLI[TLSC-1] = "else" then
        begin THENELSE := true ; TLSC := TLSC-2;
              FLI[TLI[TLSC]] := RLSC; LTF(EFLA)
        end
    else THENELSE := false
    end ;
```

```
DDEL "if" : if EFLA = 0 then RLA ; FTL(EFLA) ; EFLA := 1 ;
IF1 : OHI := 0 ; FTD ; OFLA := 1 ; goto EL ;
```

```
DDEL "then": OHI := 1 ; POP ; if THENELSE then goto DDEL "then";
        TLSC := TLSC-1 ; EFLA := TLI[TLSC-1]
        FRL (" 6T "CAC" o )
        FRL (" N 2T DECIM(FLSC) FLI ");
THEN1 ; FTL(FLSC); FLSC := FLSC+1 ; goto IF1;
```

```
DDEL "else" OHI := 1 ; POP ; if NOT then goto DDEL "else"
        FRL (" 2T DECIM(FLSC) FLI "); THENELSE;
        TLSC := TLSC+1 ; goto THEN1;
```

---

dit brengt nog verplichtingen met zich mee voor afsluiten als ;) ,] ; end

```
ADEL else : OHI := 1; POP;
        if TLI[TLSC-1] = else then begin THENELSE ;
        goto ADEL else end ;
    A: if NOT then goto A ; FRL(2T dec FLSC FLI);
        THENELSE ; TLSC := TLSC+1 ; goto THEN1;
```

tlarchive.org

if k < 1 then s > w else h ≤ c

ID	DL	TL1	RL1	FLSC
	if	if †		0
k	<	< †	↷k	
1	then		↷1	
			<	
			?	
		0 † N	→ 0	1
		then †		
s	>	> †	↷s	
w	else		↷w	
			>	
			⇒ 1	
		1 † 0⇒		2
		else †		
h	≤	≤ †	↷h	
c			↷c	
			≤	
		1→		

t-1carchive.org

if S<0 ∨ P≤Q then AA: begin if q<v then a:=v/s else y:=2*a end
else if V>S then a:=V-q else if V>S-1 then goto S

ID	DL	TLI	RLI	FLSC
	if	if †		0
S	<	< †	↷S	
0	∨		↷0	
		∨ †	<	
P	≤	≤ †	↷P	
Q	then		↷Q	
			≤	
			∨	
		0	?	
		then	N →0	1
	begin	begin †	AA→:	
	if	if †		
q	<	< †	↷q	
v	then		↷v	
			<	
		1 †	?	
		then †	N →1	2
a	:=	:=†	↷#a	
v	/	/ †	↷v	
s	else		/ s	
			:=	
		2 †	⇒2	3
		else†	1:	
y	:=	:=†	↷#y	
2	*	* †	↷2	
a	end		* a	
			:=	
	else	3 †	2: ⇒3	4
		else†	0:	
	if	if †		
v	>	> †	↷v	
s	then		↷s	
			>	
			?	
		4 †	→4	5
		then †		

ID	OL	TLI	RLI	FLSC
a	:=	:= †	↷#a	
v	−	− †	↷v	
q	else		− q	
			:=	
		5 †	⇒5	6
		else † 4:		
	if	if †		
v	>	> †	↷v	
s	−	− †	↷s	
1	then		− 1	
			>	
			?	
		6 †	N →6	7
		then †		
	goto			
S	;		⇒S	
		3:5:6:		

tl-archive.org

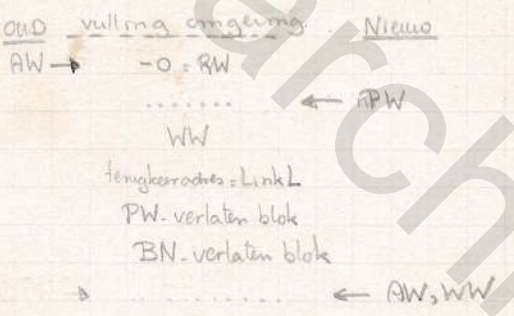
Een uitgewerkt voorstel voor de FOR-statement. AL29. 11-4-'66

Het hoofdprogramma begint via ETMP de statement Σ als procedure aan te roepen; het aantal parameters is = 0.

Het objectprogramma krijgt de beschikking over een battery subroutines, waarvan de aanroepen aangegeven zullen worden als FOR 0, FOR 1, FOR 2 etc.

De resultaatwijzer, die door ETMP op -0 gezet wordt, fungeert als "step-marker"; het terugkeeradres speelt de rol van Link L, waarmee de element list wordt afgewerkt.

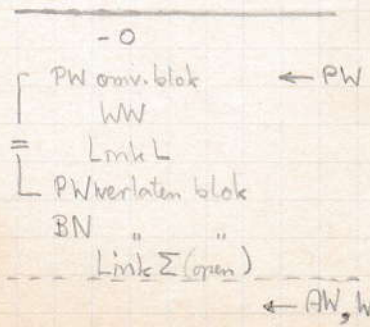
In de statement Σ arriveert de besturing met in de stapel



In de statement Σ wordt onmiddellijk FOR 0 aangeroepen.

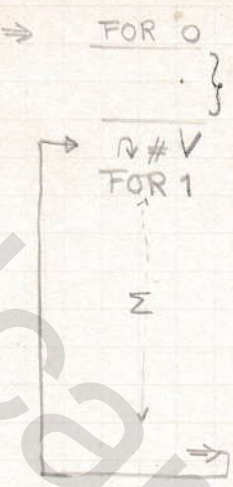
FOR 0: AW := WW := WW + 1; dus: reservering ruimte link Σ
[B] := BN + 1 ; SCC ;

na de aanroep van FOR 0 is dus het stapelbeeld



tlcarchive.org

Als V the controlled variabel is, is de structuur van de Σ-procedure als volgt:



Het FOR-complex kan over RW (voormijn part Repeat Wyzer), link L en link Σ beschikken onder controle van TPW.

FOR 1 hebbe het netto effect, dat de volgende waarde van de controlled variable ter plaatse wordt geborgen. Totaal wordt AW dus weer met 4 afgelaagd.

FOR 1 begint met link 1 (dus de hem meegegeven link) in link Σ te copieren.
Als RW=0 is (wat aanvankelijk altijd het geval is) springen we direct via link L. anders wordt de inhoud van de jongste acc. (=#V) in de volgende gecopieerd, AW := AW+4 en tenslotte TAR, voordat we via link L wegspringen.

De for list elements staan als EDSAC-program-parameters volgend op de aanroep via ETMP, en volgorde, en wel:

a) "Arithmetic expression", E_1, Het lijstelement komt in het object-programma als

⇩E_1
FOR 2 ⇒

FOR 2 Hierbij wordt link 2 in Link L gecopieerd, om te zorgen, dat straks het volgende lijstelement aan de beurt is; na :=, waardoor AW met 8 verlaagd wordt, wordt via link Σ doorgewerkt.

t-lcarchive.org

b) "While element" , E_2 <u>while</u> B, krygt in die lyst die struktuur

lnk L → ↷ E_2
FOR 3
↷ B.
FOR 4

FOR 3 We voeren uit de operatie := (omdat die uitwerking van B die nieuwe waarde van V nodig zou kunnen hebben) en verhogen AW := AW+4 om #V in veiligheid te stellen (voor het geval B = <u>false</u>): we gaan door via lnk 3.

FOR 4 We doen de operatie CAC (="?"), waarmede die boolean in de conditie komt. Zo <u>true</u>, dan vervolgen we na AW := AW−4 (weg met #V) via link 5; zo <u>false</u>, dan wordt link 4 in link L gecopieerd en springen we via link L.

c) "Step-until-element" , E_3 <u>step</u> E_4 <u>until</u> E_5, krygt in de gedaante

lnk → ↷ E_3
FOR 5
lnk → ↷ E_4
FOR 6
↷ E_5
FOR 7

FOR 5 De link 5 wordt in link L overgenomen en vervolgens wordt via lnk L gesprongen.

FOR 6 Als RW = 0 wordt AW met 4 afgelaagd, anders wordt de operatie + uitgevoerd. Daarna wordt := uitgevoerd (omdat E_5 de nieuwe waarde van V zou kunnen willen gebruiken); dan AW := AW+8 (om #V en V beide in veiligheid te brengen) en RW := ±1 afhankelijk van het teken van de 1ste vrije accumulator (sgn E_4) bovendien is RW ≠ 0, ter indicatie dat we het step element goed en wel in zijn; we gaan door via link 6.

r-larchive.org

FOR 7

We beginnen met de operatie "−" indien de jongste accumulator nu =0 is, of tegengesteld van teken als RW, dan lagen we AW met 8 af en gaan via link Σ door; anders wordt AW met 4 afgelaagd (om $V\text{-}E_5$ te verwijderen en #V tellenden), RW:=−0; en link 7 wordt naar link L getransporteerd en via link L verder gewerkt.

FOR 8

(DO) is geen subroutine, maar een inspringpunt. In een van de registers krijgt het mee, waar de berekening na de for-statement voort gezet moet worden. Dit wordt in Link L gecopieerd en de handeling "RETURN" wordt uitgevoerd.

nl(5):

ETM ⇒ met [A]=0 en [B] = α

[for 1st element
["
["
for-list

[S] := β
FOR 8 ⇒
α: FOR 0
→ γ: R#V
FOR 1
Σ
goto γ ⇒
⇒ β:

beter in mat form

tlcarchive.org

64 4P 5A

5

2A
4P
2B
2A
6A
2B
2T

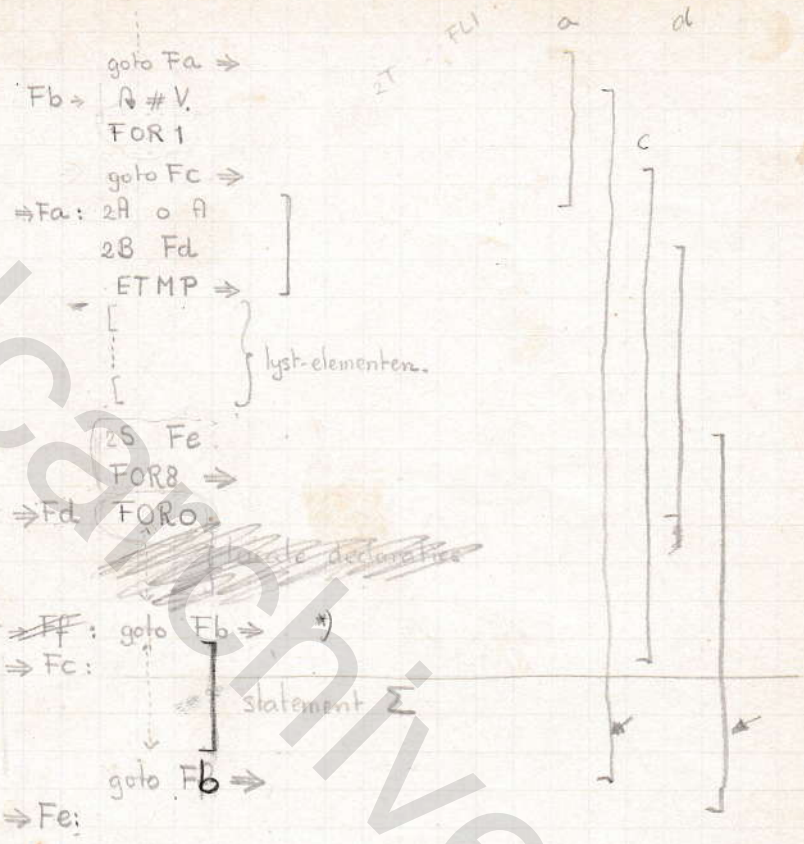
⇒.

64
36
80
36
36
64
36
64
64
64

545

t-1carchive.org

Preciesere structuur van object programma wat betreft anonyme labels bg de FOR-statement.



Opmerkingen Alleen de waarde van Ff en de FLSC van Fe hoeven in de stapel, "onder de do", Fa, Fb, Fc en Fd kunnen, als speciale werkruimten van de forclause vertaler, op vaste plaatsen staan, en wel Fa en Fd nog op dezelfde.

Verwijt: Het wordt moeilijk, om aan het einde van de declaratie, de verplichting Ff, goto Fb en Fc uit te voeren. Want hoe weten we, dat we dan aan de 1ste statement van een Σ toe zijn? Dit is alweer een dag, die op do opgezet moet worden, en die bij elke statementbegin of einde moet worden getest.

Verbeter.: Fb neemt de rol van Ff over; Fb daardoor in TLI.

t-lcarchive.org

Werkruimtes voor FOR-translation. Exclusief:
FORA en FORC

DDEL "for":

```
FRL ("2T DECIM (FLSC) FLI");
FORA := FLSC ; FLSC := FLSC + 1;
FTL (RLSC); VFLA := 1; BN := BN + 1;
OHI := 0 ; FTD; goto EL;
```

t-1carchive.org

OFLA:=1 AL35 14-4-60

```
DDEL "while": (OH1:=1; POP; if THENELSE then goto DDEL"while";
    FRL ("6T" FOR3"?");
    OH1:=0; FTD; goto EL;
```

t-1carchive.org

```
DDEL "step": OFLA:=1;

    OHI:=1; POP; if THENELSE then goto DDEL "step";
     FRL("6T" FOR5 "?"); goto EL;
```

```
DDEL "until": OFLA:=1;

    OHI:=1; POP; if THENELSE then goto DDEL "until"
     FRL("6T" FOR6 "?");
     OHI:=0; FTD; goto EL;
```

```
DDEL "do": OFLA:=1;

    OHI:=1; POP; if THENELSE then goto DDEL "do";

    if TLI[TLSC-1]="for" then FRL("6T "FOR2" ?") else
    begin TLSC:=TLSC-1;
          if TLI[TLSC]="while" then FRL("6T "FOR4" ?") else
                                    FRL("6T "FOR7" ?")

    end;

    TLSC:=TLSC-1; VFLA:=0;
     FRL("2S DECIM(FLSC) FLI");
     FTL(FLSC); FLSC:=FLSC+1;
     FRL("2T "FOR8" A")
     FLI[FORA]:=RLSC;
     FRL("6T "FORO" ?");
     FRL("2T DECIM(TLI[TLSC-2]) RLI A");
     FLI[FORC]:=RLSC;
EFLA:=OHI:=0; FTD; goto EL;
```

RLSC F6
FLSC Fe
do

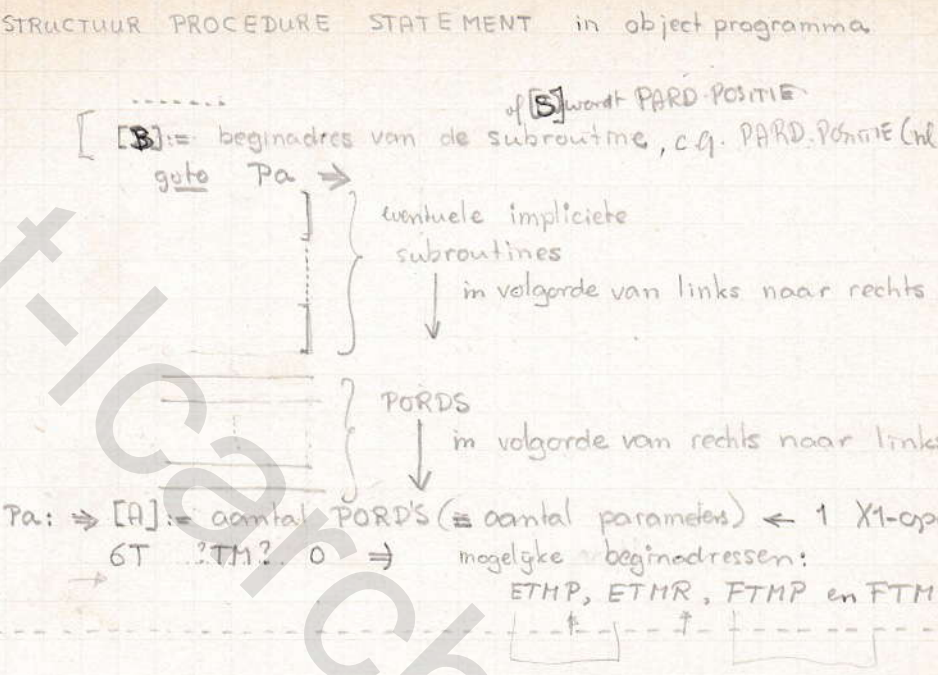
tcarchive.org

```
DDEL";" :

    OFLA:=OHI:=1; POP; if THENELSE then goto DDEL";";

    if TLI[TLSC-1]="do" then

begin TLSC:=TLSC-3; BN:=BN-1;
      FRL("2T DECIM(TLI[TLSC]) RLI A");
      FLI[TLI[TLSC+1]]:= RLSC; goto DDEL";"
end;

if SFLA=1 then

begin SFLA:=0;

KLAAR1: if TLI[TLSC-1]="switchkomma" then
        begin TLSC:=TLSC-2
              FRL(2T DECIM(TLI[TLSC])RLI A); goto KLAAR1
        end;
        TLSC:=TLSC-1;          vervolgens de :=
        LTF(ID);
        LDEC;                  (LABELDECLARATIE)
        FRL("1T 16 X1");
        TLSC:=TLSC-1
        FLI[TLI[TLSC]]:=RLSC
end;
```

t-carchive.org

Een uitgewerkt voorstel over PROCEDURE STATEMENTS. AL38 19-4-'60

STRUCTUUR PROCEDURE STATEMENT in object programma



Aan het subroutinecomplex wordt een speciale ingang toegevoegd, nl. EIS : End of Implicite Subroutine ; dit programma probeert de inhoud van de jongste accumulator aan de eigen identifier toe te kennen (dus afhankelijk van RW dus.); dan naar RETURN. Als in een parameter een procedure met parameters staat, wordt deze altijd in de R-versie aangeroepen: je kan nooit weten!

We moeten drie soorten actuele parameter onderscheiden

A expressie probeert resultaat af te leveren
B loze identifier geen gegevens verwachten, wel impl. subroutine, behalve als designational
C enkele gedimensioneerde variabele. probeert adres af te leveren, tenzij designational

Als we distribueren op het sluitsymbool van de parameter

Als TLI niet eindigt op "," } type A
of NFLA en AFLA zijn beide = 0

Als TLI wel eindigt op , maar NFLA = 1 type B
(NFLA = 0, dus) AFLA = 1 type C

t-lcarchive.org

PROCEDURE STATEMENT AL 39 19-4-60

Als we aan een parameter gaan beginnen, ligt er in TLI: RLSC " " ,

om te weten, waar een eventuele impliciete subroutine begonnen is; als een parameter is afgewerkt, wordt de RLSC in TLI vervangen door een beschrijving van de PORD, die dan gemaakt kan worden.

Nu de identificatie van het type actuele parameter iets vollediger.

1° Als TLI niet eindigt op "," dan is het zonder meer een expressie. (een som, of een product of zo), type A.

2° Als TLI wel eindigt op "," maar NFLA = AFLA = 0, wat zijn dan de mogelijkheden.
   a) een ding tussen haakjes
   b) een procedure met parameters

3° Als TLI eindigt op , maar NFLA = 1
   a) loze identifier } ik moet onderscheiden: de formele label
   b) switch-element. (AFLA=0) } deze moet ik er uit peuteren. en het formele switch element

4° Als TLI eindigt op , maar NFLA = 0, dan is AFLA = 1 en hebben we de subscripted variable.

Er is iets voor te zeggen, om bij de switch wel ID te reconstrueren, maar met de NFLA

rtcarchive.org

AL 40 19-4-'60

EFLA =1 Expressie-situatie
=0 non-expressie-situatie.

DE expressievlag EFLA beheerst, of procedures in P of R versie aangeroepen moeten worden, nl. P als EFLA=0, R als EFLA=1

EFLA wordt op 1 gezet door :=, ook door := in for clause
op 0 gezet door ;, begin, do

EFLA wordt op 1 gezet door if, nadat die oude in de stapel is gezet; die komt er bij then weer uit; bij else moeten we het weer halen!

EFLA wordt op 1 gezet bij [bij array element en switch element.

EFLA wordt op 1 gezet bij (parameter haak, nadat oude in de stapel; komt bij) er weer uit.

NB. EFLA =1 sluit niet uit, dat er een statement komt!
EFLA =0 sluit wel uit, dat er een expressie komt.

tlarchive.org

```
procedure POP ;
begin if NFLA = 0 then
      begin if AFLA = 1 then
            begin AFLA := 0 ; FRL ("6T "TAR" 1) end
      end else
      begin if PFLA = 1 then
            begin BPR ; FRL ("2A 0 A")
                  if EFLA = 0 then
                  begin if FFLA = 0 then
                        FRL ("6T "ETMP" 1") else
                        FRL ("6T "FTMP" 1 )
                  end else
                  begin if FFLA = 0 then
                        FRL ("6T "ETMR" 1") else
                        FRL ("6T "FTMR" 1 )
                  end ;
POP2:             AFLA := NFLA := 0 ;
                  goto POP1
            end ;
            if JFLA = 1 then
               begin if FFLA = 1 then
                     begin BPR ; FRL ("6T "TFR" 1")
                     end else
                     begin POPA := BN(ID); if POPA ≠ BN then
                           begin FRL ("2B DECIM(POPA) A");
                                 FRL ("6T "GTA" 0")
                           end ;
                           PSOID
                     end ;
                     goto POP2
               end ;
```

PTM (PRODUCEER TRANSMARK)

ie KOMMA

```
AVR ;
if OHI ≤ OHIT(TLSC-1) ∧ OPA(TLSC-1) then
begin "Fill RLI met adreshebbend + - * of /, Integer, Reeel of Formeel; Statisch of dynamisch
e TLSC := TLSC - 1
end else
begin "Fill RLI adreshebbend met ↑, I, Rel F, St of Dyn, "end ;
goto POP2 ;
```

t-larchive.org

```
POP1:  if OHI ≤ OHIT (TLSC-1) then
         begin "Fill RLI met adresloze operatie"; TLSC := TLSC-1; goto POP1
         end ;
end ;
```

t-lcarchive.org

OUTPUT CODE

Output in RLI bestaat uit een binair woord + 7 bits, die in principe, een additieve constante bepalen.

AL43 27-4-'00

0	XO			
1	RLI			
2	FLI			
3	KLI			
4				
5				
6				
7				
8	6T	"ETMR"	1	EXTRANSMARK RESULT
9	6T	"ETMP"	1	EXTRANSMARK PROCEDURE
10	6T	"FTMR"	1	FORMTRANSMARK RESULT
11	6T	"FTMP"	1	FORMTRANSMARK PROCEDURE
12	2T	"RET."	A	RETURN
13	2T	"E/S"	A	END OF IMPLICIT SUBROUTINE
14	6T	"TRAD"	1	TAKE REAL ADDRESS DYNAMIC
15	6T	"TRAS"	1	TAKE REAL ADDRESS STATIC
16	6T	"TIAD"	1	TAKE INTEGER ADDRESS DYNAMIC
17	6T	"TIAS"	1	TAKE INTEGER ADDRESS STATIC
18	6T	"TFA"	1	TAKE FORMAL ADDRESS
19	6T	"FOR0"	0	FOR0
20	6T	"FOR1"	0	FOR1
21	6T	"FOR2"	3	FOR2
22	6T	"FOR3"	3	FOR3
23	6T	"FOR4"	3	FOR4
24	6T	"FOR5"	3	FOR5
25	6T	"FOR6"	3	FOR6
26	6T	"FOR7"	3	FOR7
27	2T	"FOR8"	A	FOR8
28	6T	"GTA"	0	GOTO ADJUSTMENT
29	6T	"SSI"	1	STORE SWITCH INDEX
30	6T	"CAC"	0	COPY BOOLEAN ACC. INTO CONDITION
31	6T	"TRRD"	1	TAKE REAL RESULT DYNAMIC

t-lerchive.org

OUTPUTCODE

AL44 27-4-60

```
32    6T  "TRRS"  1   TAKE REAL RESULT STATIC
33    6T  "TIRD"  1   TAKE INTEGER RESULT DYNAMIC
34    6T  "TIRS"  1   TAKE INTEGER RESULT STATIC
35    6T  "TFR"   1   TAKE FORMAL RESULT
36    6T  "ADRD"  2   ADD REAL DYNAMIC
37    6T  "ADRS"  2   ADD REAL STATIC
38    6T  "ADID"  2   ADD INTEGER DYNAMIC
39    6T  "ADIS"  2   ADD INTEGER STATIC
40    6T  "ADF"   2   ADD FORMAL
41    6T  "SURD"  2   SUBTRACT REAL DYNAMIC
42    6T  "SURS"  2   SUBTRACT REAL STATIC
43    6T  "SUID"  2   SUBTRACT INTEGER DYNAMIC
44    6T  "SUIS"  2   SUBTRACT INTEGER STATIC
45    6T  "SUF"   2   SUBTRACT FORMAL
46    6T  "MURD"  2   MULTIPLY REAL DYNAMIC
47    6T  "MURS"  2   MULTIPLY REAL STATIC
48    6T  "MUID"  2   MULTIPLY INTEGER DYNAMIC
49    6T  "MUIS"  2   MULTIPLY INTEGER STATIC
50    6T  "MUF"   2   MULTIPLY FORMAL
51    6T  "DIRD"  2   DIVIDE REAL DYNAMIC
52    6T  "DIRS"  2   DIVIDE REAL STATIC
53    6T  "DIID"  2   DIVIDE INTEGER DYNAMIC
54    6T  "DIIS"  2   DIVIDE INTEGER STATIC
55    6T  "DIF"   2   DIVIDE FORMAL
56    6T  "IND"   1   INDEXER
57    6T  "NEG"   0   INVERT SIGN ACCUMULATOR
58    6T  "TAR"   1   TAKE RESULT
59    6T  "ADD"   2   ADD
60    6T  "SUB"   2   SUBTRACT
61    6T  "MUL"   2   MULTIPLY
62    6T  "DIV"   2   DIVIDE
63    6T  "IDI"   2   INTEGER DIVISION
```

tfcarchive.org

```
64    6T  "TTP"  ?    TO THE POWER ↑
65    6T  "MOR"  ?    MORE >
66    6T  "LST"  ?    AT LEAST ≥
67    6T  "EQU"  ?    EQUAL =
68    6T  "MST"  ?    AT MOST ≤
69    6T  "LES"  ?    LESS <
70    6T  "UQU"  ?    UNEQUAL ≠
71    6T  "NON"  ?    NON ¬
72    6T  "AND"  ?    AND ∧
73    6T  "OR"   ?    OR ∨
74    6T  "IMP"  ?    IMPLIES ⊃
75    6T  "QVL"  ?    EQUIVALENT ≡
76    6T  "abs"  ?
77    6T  "sign" ?
78    6T  "sqrt" ?
79    6T  "sin"  ?
80    6T  "cos"  ?
81    6T  "arctan" ?
82    6T  "ln"   ?
83    6T  "exp"  ?
84    6T  "entier" ?
85    6T  ST     1    STORE
86    6T  STA    2    STORE ALSO
87    6T  STP    1    STORE PROCEDURE VALUE
88    6T  STAP   0    STORE ALSO PROCEDURE VALUE.
89    6T  "SCC"  0    Short CirCuit
90    6T  "RSF"  0    REAL ARRAYS STORAGE FUNCTION FRAME
91    6T  "ISF"  0    INTEGER  "  "  "  "  "
92    6T  "RVA"  0    REAL VALUE ARRAY  "  "  "  "
93    6T  "IVA"  2    INTEGER  "  "  "  "  "
94    6T  "LAP"       LOCAL ARRAY POSITIONING
95    6T  "VAP"       VALUE ARRAY POSITIONING
```

t-carchive.org

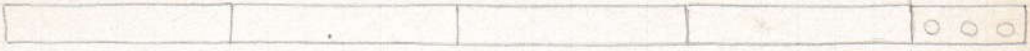
OUTPUTCODE

```
96                      START
97                      STOP.
98   TFP            .   TAKE FORMAL PARAMETER.
99   TAS
100  OTBC6.
101  ------             FLOATER.
102  read
103  print
104  TAB
105  NLCR
106  XEEN
107  SPACE
108  stop
```

ti-archive.org

NAAM

1e 1-woords-naam : aantal symbolen ≤ 4



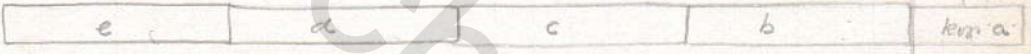
a:	a	o	o	o	o
ab	b	a	o	o	o
abc	c	b	a	o	o
abcd	d	c	b	a	o

---

2 woords naam: begint met abcde

kop woord bestaat uit

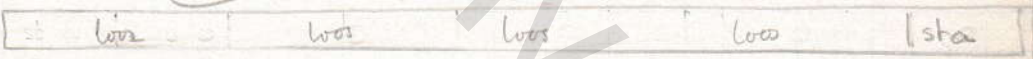
kop a ≠ o



staart woord.

abcde

(woo = 63)



abcdef

	woo	woo	woo	f	sta
fg	woo	woo	g	f	sta
fgh	woo	h	g	f	sta
fghi	i	h	g	f	sta

Eenwoordsnaam: INW ID →

Inhoud naamlijst ↑ o

Tweewoordsnaam FNW INW ID →

↑ aflopend nummer

t-lcarchive.org

Lower Case	Upper Case		Bandwidth
1	V	17	1
2	*	18	2
3	/	3	19
4	=	20	4
5	;	5	21
6	[	6	22
7	]	23	7
8	(	24	8
9	)	9	25
0	Λ	48	32
a	A	113	97
b	B	114	98
c	C	99	115
d	D	116	100
e	E	101	117
f	F	102	118
g	G	119	103
h	H	120	104
i	I	105	121
j	J	65	81
k	K	66	82
l	L	83	67
m	M	68	84
n	N	85	69
o	O	86	70
p	P	71	87
q	Q	72	88
r	R	89	73
s	S	34	50
t	T	51	35
u	U	36	52
v	V	53	37
w	W	54	38
x	X	39	55
y	Y	40	56
z	Z	57	41

Flexowriter code

L.	U.		band W.
+	"	96	112
?	?	75	91
	'	43	59
•	:	123	107
-	⌐	80	64
—	|	30	14
<	>	33	49
	Upper		124
	Lower		122
	TAB	46	62
	NLCR	10	26
	Punch off		47
	Punch on		0
	space	0	16

68 ∴ U2 ≠

69 ↑ U3 ≯

71 ≥

73 ≤

75 ∔

79 ¬

80 ≡

92 :=

93 spare

tclarchive.org

ID

1° Getal of Variabele.

2° Als Getal, dan plaats in KL1, + integer of real.

3° Als Variabele.
1 FORMEEL of Niet.

4° Als FORMEEL, dan blokenummer en positie tov. PW.
Als non formeel:
dan integer of real,
statisch of dynamisch
blokenummer en positie tov. PW.
dan nulpunt := 0.

NL1 characteristicum: adresweerd uit de naamlyst.

$d_{14} - d_0$: bepalen by dynamisch gegegeven positie $\times 2^{10} + bn$.
by statisch gegeven adres tov. van nulpunt.

d_{15} =1 nulpunt dynamisch
=0 nulpunt statisch.
d_{16} = FFLA
d_{17} = JFLA =1 Label en Switch.
d_{18} = PFLA =1 Procedure.
d_{19} = 18 = Integerbit
d_{20} =
21 =
22 =
23 =

bloknummer by label en switch;
by proc blokenummer -1 van de body

d_{24} } coden voor de vier vaste nulpunten. / by statisch
d_{25} } 0=X0 ; 1=RL1 ; 2=FL1 ; 3=KL1. } adres.

als $\begin{cases} d_{17} = d_{15} = 1 \\ d_{16} = 0 \end{cases}$, dan $d_{14} - d_0$ ~~NLSC van het ID in naamlyst.~~
d_{24} en d_{25} dan =0

d_{26}=1 d_{25}=0 nog te verwerken value array.

d_{26}=1 d_{25}=1 nog te reserveren locale arrays.

t-1carchive.org

bij een switch bergen we in TLI

op verschijn vd :=

→ plaats van ID in naamlijst.

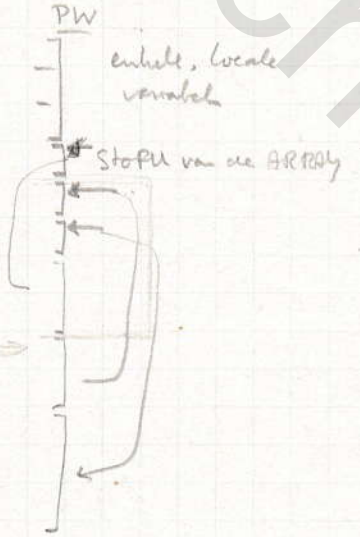
FLSC
Switch
RLSC
switchkomma

produceer 2T FLSC FLI
en Lnm FLSC op.

MFLA = 0
EFLA = 0
IFLA = 0.

1° detectie blok-begin.
2° structuur object programma bij locale arrays.
3° gedaante vd stofu.

Procedure (,a1,) value a
array a.



tl-archive.org

Declaraties: alleen na begin

nonsens { alle integers · identifiers worden gelezen; te worden ~~PW op volgorde~~ op volgorde ter PW gelocaliseerd.
alle reals dito. als dit klaar is, weten we, met hoeveel het objectprogramma AW en MW verhogen moet: ik stel voor, dat we dat in het programma doen.

~~Nu scann~~

Nu ~~de locale arrays~~:

~~We scannen de locale arrays; reserveren plaats in de Mapel en noteren dit in~~

We werken de subscript-bounds uit voor 1 locale array: we roepen aan de subroutine Maak STOFU, die alles behalve de positie invult.

EN ZO VOORT.

EN DAN GEADRESSEERD DE RESERVEER-ROUTINE Aanroepen.

$$\Delta_1 i_1 + \Delta_2 i_2 + \Delta_3 i_3$$

$$S_0 (i_1 + S_1 (i_2 + S_2 i_3))$$

1 v/2 range van de i^{de} index.

Voor value array wordt een royale stofu-ruimte in de stapel gereserveerd. (vaste);

Aanroep: Prepareer STOFU VALUE ARRAY; deze krijgt de PARD-positie mee, en transporteert onder controle van AW het physisch adres van de buiten-stofu (~~descriptor~~) (ook al de vaste gegevens van de stofu. Als deze aanroep gesproken is, wordt in NLI de identID op de locale STOFUPLANTS gezet.

Verder de aanroep: Reserveer en initialiseer; deze ~~begint met AW passend op te hogen~~ vult de waarden in te beginnen bij AW; vult de stofu passend in en transporteert de waarden AW en MW.

Reserveer en initialiseer moet voor de andere arrays.

Archive.org

De volgorde in het object programma

~~TIAD - TRAD~~

Voor value variabele:

2S pardpositie A
TIAD of TRAD
2S pardpositie A
TFR
ST

En halen de indicatie FORMAL uit NL weg: hebben inlezen dat daar al neergezet.

2A PFLA Z
N 1A FFLA Z

Nab. PFLA = 1.
FFLA = 0.

TFO.

N
N 8B
1 2A 8 2D

⇒

FRADE
FRDM
FRDM
FRDME

NWaAW →

t-lcarchive.org

transport TLI → NLI inlassen bij begin blocks en ook in DDEL "do"
afvallen NLI inlassen aan einde blocks, b.v. in "DOT".
we vormen in het ~~blokbegin~~ ~~TLI-symbool~~ begin *.
TLI-symbool BB

---

subscriptbounds
$[l_0 : u_0, l_1 : u_1, \ldots [l_{(n-1)} : u_{(n-1)}]$.
begin adres vh. array.
begin adres vh array $- \sum l_i \Delta_i$.

Δ_0 1 of 2 afh. van IB.
Δ_1 $(1+u_0-l_0) \times \Delta_0$
Δ_2 $(1+u_1-l_1) * \Delta_1$
Δ_{n-1} $(1+u_{n-2}-l_{n-2}) * \Delta_{n-2}$

} als we het zo doen, vormt indexer een scalair product

Δ_n $(1+u_{n-1}-l_{n-1}) * \Delta_{n-1}$: totale lengte vh. array.

$\sum_{i=0}^{n-1} l_i \Delta_i$ dit is nl. de bydrage., tgv.

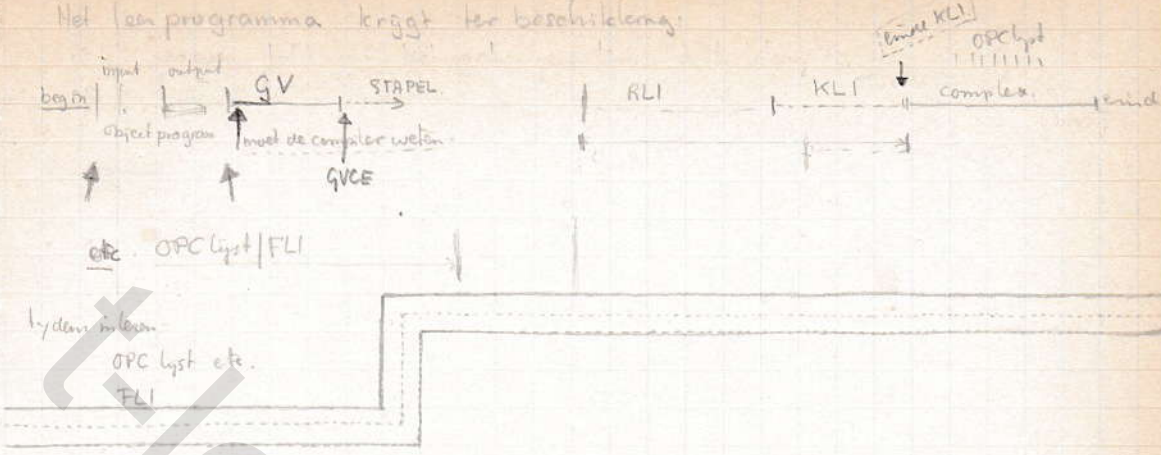
STOFU: → begin adres vh. array WW
→ begin adres vh array $- \sum_{i=0}^{n-1} l_i \Delta_i$ WW1
$\Delta_0 = 1$ of 2 WW2
$\Delta_1 = (1 + u_0 - l_0) * \Delta_0$

$\Delta_{n-1} = (1 + u_{n-2} - l_{n-2}) * \Delta_{n-2}$

$\Delta_n = -(1 + u_{n-1} - l_{n-1}) * \Delta_{n-2}$. WW1&m

tlarchive.org

Het (een programma krygt ter beschikking:



Compiler produceert een band voor bepaalde installatie, dit in verband met GVC. De beginstelling hiervan zal afhangen van de installatie, waar het objectprogramma op moet draaien.

Deze constante kan je dood geheugen hebben, maar dat is verwerpelijk. Dus komt deze constante in levend geheugen.

Dus zelfs, als we de start processen ~~auto~~ met via band doen, moet toch, als de compiler in dood geheugen zou staan, moet toch een ~~test~~ preparatiebandje ingelezen worden.

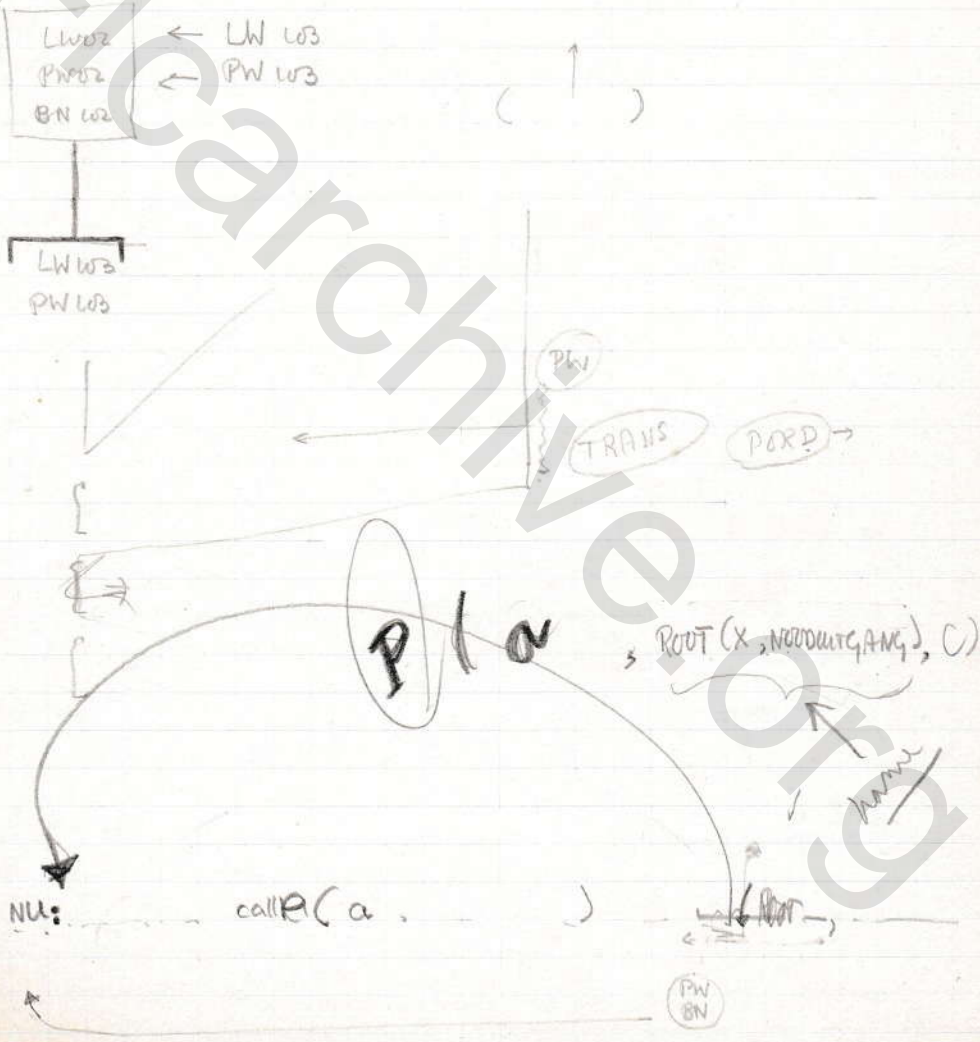
tilarchive.org

```
16 X1   Switch Index
17 X1   BW
18 X1   AW
19 X1   PW
20 X1   BN
21 X1   GVC E.
22 X1   BIB6  Begin Input Buffer Class 6
23 X1   BOB6  Begin output Buffer Class 6.
```

tl-carchive.org

De functie van de PW-ketting is voor het vinden van de globale variabele; zodra nu. we een goto uitgang uit een procedure tegenkomen, dan is dat tot op zekere hoogte een globale variabele. dat is het. En via de PW-ketting in het lokalenummer descriptief, mits ik er maar voor zorg, niet in een anonym tussen niveau te trappen. & PW-keten is dan lekker langer!

Dus nu het plaatje voor de anonyme subroutine



tt-archive.org

bij elke onder opgave v BN
bij de expliciete verlaatsprong: [formele verlating]

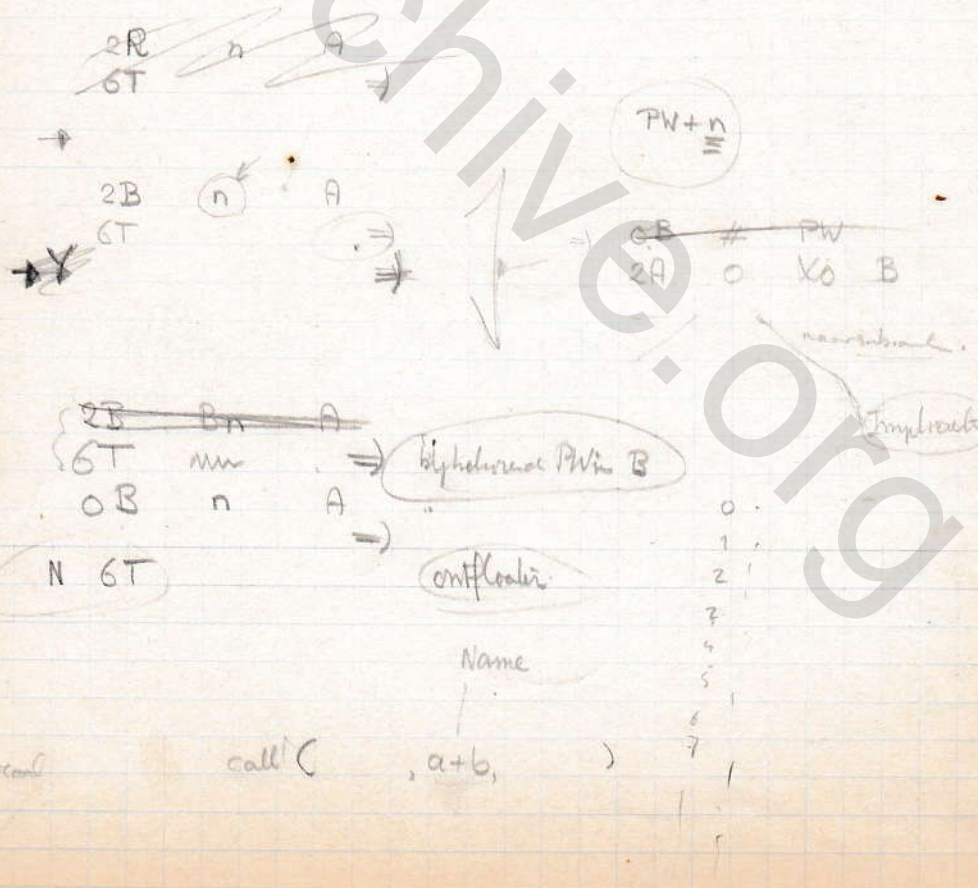
1x PW - geeft PW en BN geeft PW en BN

draai LW, terug

(call A (B) B)

a x b

A[a, P, t] B[b].



t-lcarchive.org

Opspoelers 2 (Ponskamer . X1). van Lelmen

AFLOOP BAKJES 3 of 4. (X1, Reproductieapparatuur; (breed). Flexowriter, 7gats reproductie) van Lelmen

7-gatsreproductie apparatuur. EL-schema vraag; van Lelmen solderen. Pilkes apparatuur

Pin-feed papier X1-schrijfmachine. W-duizend 2-wegdraging. ~~W-duizend~~ van Lelmen

X1 - 1 (~~en~~ 0 of O)

Subroutinereproductie.

Editing — ~~Inks~~ Barning.

Drukker. — ~~Selectr.~~

Onze Dood-gekkengenbestelling. P9. etc. ik zal voorstel maken. II

ARCHIEF - STANDAARD - BANDEN. zal 't opdragen aan Marlene - Fiek.

Maken van standaard assemblage van P9. Dekken Barning

Documentatie MCX1 - specificatie. Stencil Dekken Barning

Meubilair X1-kamer. 4 1 tafel als b.v. one zolfde blauwe stoel zonder armleuning. Emmeln. Schmidt

Programmeercursus ? Nilmy Cursus Aanvraag

ALGOL-Cursus: Syllabus

Manie. weg. termijn. 1 July ~~Schmidt~~

~~Overdatie Boodel~~

Elgarchive.org

NSS INTERN SUBR. NEXT SYMBOL. INTO S HTO 16-6-'60

Aanroep 6T 0 H To 14 ⇒ :

Voorlopige versie:
text band: ruffel blanks.
symbolen in 2 pentaden (binair)
aan het einde een D,
X aan het begin wordt geschipt.
14ZE2=0: we skippen blanks aan het begin.
14ZE2=31, we lezen.
14ZE2<0, we zijn klaar.

```
        DA 0 H To   DI
⇒  0    2A 22  X0
   1    3S 14 Z E2  Z
   2  N 2T 7 H To A      →
5→ 3    6T 15  Do 14  ⇒  LWW
   4    3S 24  X0   Z
   5  Y 2T 3 H To A      →
14 → 6    7S 14 Z E2
2 → 7     4P  SS  P
   8  Y 2T 12  Do A      →   Exit LP
12 → 9    6T 15  Do 14  ⇒  LWW
   10     2S 24  X0
   11   U 1LS 31      A Z
   12 Y 2T 9 H To A      →
   13   U 1S 3        A P
   14 Y 2T 6 H To A      →
   15    0P 5  SS
   16    6T 15  Do 14  ⇒  LWW
   17    0S 24  X0
   18    2T 11  X2   E ⇒
```

5-gatband

tclarchive.org

PONSVOORSCHRIFT

ROFFEL BLANK, gevolgd door X (= ERASE)

opeenvolgende ALGOL-symbolen in twee pentades
nl. 1ste pentade (0, 1, 2, 3) geeft aan 32-voud
2de pentade geeft aan de eenheden.

NB. Aan het begin van elke ALGOL-woorden extra pentades X (= ERASE) geskipt.

Het einde van de band markeert men overeen ~~(een pentade D (nl. 4 ≤ endpentade ≤ 30))~~

roffeltje blanke XDO : roffel blanks.

t-lcarchive.org

```
begin real a, x; a := x := 7 end
```

Geproduceerd op 18-6-60:

```
      BIN W          OPC
0 KLI   0            96        START
1     2B  128   A      0
2         0           15        TRAS    ] Q # a
3     2B  130   A      0
4         0           15        TRAS    ] Q # x
5     2B  0     A      3        KLI.     ] Q 7
6         0           34        TIRS
7         0           26        STA      :-
8         0           25        ST       :=
9         0           27        STOP
0 KLI     7            0
```

```
          10            0        RLSC E
           1            0        KLSC E
           0            0        TLSC E
         132            0        QVC E
```

tlcarchive.org

16-6-60 TEST 2

begin integer GEHEEL; Niks: GEHEEL := -12.5 $_{10}$ 1; goto Niks end

---

Gproduceerd op 19-6-60

```
                    OPC
                    96      START
1)  2B  128  A
                    17      TIAS
    2B   0   A      3       KLI
                    32      TRRS
                    57      NEG
                    85      STORE
    2T   1  X0  A   1       RLI
                    97      STOP
-------------------------------
  125 x 2^-7                125
  2^11 + 7
-------------------------------
     9                      RLSC E
     2                      KLSC E
     0                      PLSC E
    129                     GVC  E.


Niks          00 00 01
```

t-larchive.org

```
begin real x ; integer i ;
    x:=0 ;
    for i:= 1 step-2*i +(if i>0 then -1 else 1) until i do
    begin x:= x + 1/i ; if i>1000 then goto KLAAR end ;
KLAAR: end
```

tlcarchive.org

```
integer x, n, k;
begin integer x;          (must go)

    real procedure SIGMA (i, k, l, t); value l;
        integer i, k, l; real t;

    begin real x;   x := 0 ;

        for i := k step 1 until l do x := t + x ;

        SIGMA := x

    end SIGMA ;

    x := SIGMA (n, 1, 10, SIGMA (k, 1, n, 2) - 1)

end
```

t-1archive.org

16-6-60 TEST 5

```
begin integer n;
integer procedure GGD(u1, u2); value u1, u2; integer u1, u2;
GGD := if u2 = 0 then abs(u1) else GGD(u2, u1 - u1 ÷ u2 * u2);
TERUG: n := GGD(2409, 4605); goto TERUG
end
```

4608
512
4096

tt-archive.org

```
boolean
begin P17, P289;

    boolean procedure Prime(n); value n; integer n;

    begin integer t;

        if n≠2 ∧ n-n÷2*2=0 then goto FALSE;

        if n=3 then goto TRUE; t:=1;

    M : t:=t+2; if t*t>n then goto TRUE;

        if ¬Prime(t) then goto M;

        if n-n÷t*t≠0 then goto M;

    FALSE: Prime := false; goto END;

    TRUE : Prime := true;

    END : end;

    P17 := Prime(17); P289 := Prime(289)

end
```

t-Carchive.org

compiler 601 REC

```
begin  array  x,y [-1:4];
    x[1]:= -1; y[x[1]]:= 3;
    x[-1]:= 10 7; x[0]:= -10 7;
    y[0]:= +1 10-7; y[-x[0]*y[0]]:= 15.5;
    y[2]:= +125 10-3; y[3]:= .125;
    x[2]:=00.125 ; x[3]:= -.125 10-0;
    x[4]:= 32.7680 10+3 ;y[4]:= 327.68 10 2;
end
```

tlcarchive.org

KUL 6-16-60 TEST 8

```
begin real b, c, d, B;
    integer a1, e, j;
    array a [1:7, 0:3];
    integer array i [0:10];
    integer procedure k (a1, e); real a1, e; k := p(a1)*e;

    real procedure p (B); value B; integer B;
    L: p := B * cos(a1);
    if a1 + a[i[k(a1, b*(3+ p((a[1,1]+c)*d↑
        if cos(exp(a1)) > 0 then 2 else - c * ln(e))))+j],
        j+1] * k(a[k(a1, b), j], B * cos(exp(a1))) ↑
        2↑ ln(p(a1)) * c + d = a1 * 3 ∨
        c > d ∧ ¬ d ≤ a1 ≡ sin(a1) ≤ 0 ⊃ a1 < 0
    then b := c else c := d
end
```

1680 in Basic

RCI 8½ sec
ICI ½
Rest 3sec

t-ICA rchive.org

```
comment TEST 9;                    (KUL)  16-6-'60   TEST 9
begin real a, b, c;

    real procedure KUL (v, w); real v; label w;

        begin real d;

        procedure KULI; if v > a then KUL := a := 3 else

                                    begin d := KUL := 5; goto w end;

        d := 0;
        if v ≠ 0 then KULI else

        if a > 0 then KUL := d := 7 else a := KUL := 20;

        if d ≠ 0 ∧ a = 0 then goto w

        end KUL;

        a := 1; b := c := 2;

L :     KUL (3, L)

end
```

÷	≔	≠	≠
≥	≥	⊂	⊄
≤	≤	⊃	⊅
⊃	⊐	↑	↑
≡	≡		

t-larchive.org

```
'begin begin array a,b[1:5];
    a[3]:=(.7) end end
```

t-1carchive.org

TEST 11 26-6-60

```
begin integer n; n:=1;
Telangenaam:
    begin integer k; integer array a[0:n];
M:  for k:=0, k+1 while k≤n do
        begin integer i; switch S := M, L, S[4], Telangenavel;
L:  a[k]:=0; for i:=0 step 1 until k-1 do a[k]:=a[k]+a[i];
        n:= n+1; goto S[3] end end end
```

t-lcarchive.org

TEST 12 26-6-60

```
begin integer n; array a[0:10]

    procedure KUL ( i, a1, a2, a3, p); value i, a1, a2;
      integer i; array a1, a2, a3; integer procedure p;

    begin own real s; array b,c [0:i, 0:i+1]; p(p,b,s) end

end
```

FLCarchive.org

TEST 13:

```
begin boolean b; Boolean B; real x,y;
    if b then begin x:=1; goto in end;
    if B then begin x:=-1; in: y:=2 end end
```

t-1carchive.org

Test 14 a 30-6-60

```
begin
    begin real x;
        procedure P(u,v); real u; label v;
        begin procedure Q; if u > +((0)) then goto (v);
            Q; end
        P(x,y); y:
    end
end
```

```
0                   26
1   2A  0  A
2   25  5  A  1
3                   9   ETHP
4   2T  0  X0  2 ⇒  over blok.
⇒5  2B  1  A
6                   89
7   2A  2  A
8   4A  17 X1
9   4A  18 X1
10  2T  1  X0  2 ⇒  over P
⇒11 2B  2  A
12                  89
13  2T  2  X0  2 ⇒  over Q
⇒14 2B  3  A
15                  89
16  25  5-2 A
17                  35  TFR
18  2B  0  A  3
19                  34  TIRS
20                  84
21                  65  >
22                  30
23  N ST  3  X0  2 →
```

```
28  25  7-2  A
29              35  TFR
→30             12  RET   Q
31  2B  X1  A   1   begin Q.
32  2A  0   A
33
34              9   ETHP : call Q
35              12  RET  end P
36
37  2B  11  A   1   beg= P
38  2T  4   X0  2   over par. PORDS ⇒
39  2B  1       A
40                  GTA
41  2T  5   X0  2   ⇒ -- Y
42
43                  PORDY
44  Q=2. a=32   1   PORDX
45  bnel Q=0. b=1. a=5
46  2A  2       A
47                  9   ETHP
48                  12  RET
49                  97
50                  0
```

```
KLI:    42      RLSC E
        1
        6
        28
```

```
FLI  0  41 1    3  25 -1
     1  30 1    4  38 -1
     2  26 1    5  40  1
```

t-larchive.org

```
begin real x,y,a,b,c;
    x:= a*(y:= b+c)+y   end
```

```
0                   96
1   28  128   A
2                   15  TRAS
3   28  132   A
4                   32  TRRS
5   28  130   A
6                   15  TRAS
7   28  134   A
8                   32  TRRS
9   28  126   A
10                  37  ADRS
11                  86  STA
12                  61  MULL
13  28  130   A
14                  37  ADRS
15                  85  ST
16                  97  klaar

17
0
0
138
```

t-1carchive.org

TEST 16 30.6.60

```
begin boolean B; integer n;
    array a[-3:+1,-3:+1]; real procedure Q; Q:=2.5;
    begin integer array a[-3:+1,-3:+1],
        b[-if ¬B then n+1 else +n:+if ¬B then n else 2*n,
          +if ¬B then -(n+1) else (n):+if B then n else
          if B then n↑2 else Q]; Q end end
```

t-1carchive.org

```
begin
    print (sign (entier(arctan (cos (sin(sqrt (abs (ln (exp (read))))))))))
end
```

(10×)

```
> 30  0   96
        1   102
        2   83
        3   82
        4   76
        5   78
        6   79
        7   80
        8   81
        9   84
        10  77
        1   103
        2   97
```

t-1carchive.org

```
begin integer M; M:= read;
    begin real y, X; real array a [1:M]; integer k;
    real procedure SIGMA(a, f, x, p, N); value a, N, x;
        integer N, p; real a, x; real procedure f;
        begin real s; s:= 0;
            for p:=1 step 1 until N do s:=s+ a*f(p*x);
            SIGMA := s
        end;
        y := SIGMA(a[k], sin, X, k, M)
    end
end
```

t-1carchive.org

Roffel P

TEST BAND
TEST 21.

1 T
4 T
9 SS
1S6 T.
2S5 W
SSS+8 6 T PPPPP
+SSS4S9T
+.64E2T
.81 E+2 W
ESS2 W
121 T
144 P
169 T
19600.0 E$$-$$2$$
225 S000 S000 S000 S000 S000 S000 E-18 T
- 256 P
- 289 P
0 P
+50 $ 8212 $ 1521 $ 51 $ 5868 T
+ E 604

bis.

t.lcarchive.org

TEST 20

```
280        96
   1   2T 2-8-6   →
22→2   2S 4-10  A
   3                TIAS
   4                FOR1
   5   2T 2-8-23 A
1 →6   2A  0    A
   7   2B 2-8-21 A
   8                ETM7
   9   2B 2-9-29 A          5
  10                TIRS
   1                NEC 8
   2                FOR 5
   3   2B 2-9-30 A          1
   4                TIRS
   5                FOR6
   6   2B 2-9-31 A          10
   7                TIRS
   8                FOR7
   9   2S 2-9-5  A
  20                FOR 8
→  1                FOR 0
   2   2T 2-8-2  A   ⇒
5→ 3                NLCR
   4   2B  4-10  A          X
   5                TIRS
   6                PRINT
   7
   8
```

2 X 3
6 X 3
8 X 6

98

t-1carchive.org

Test 20

```
begin integer x; real y;
L: for x:=-5 step 1 until 10 do
    begin NLCR; print(x); TAB; y:=x; print(y) end;
   for x:=1 step 1 until 10000 do; goto L end
```

Test 21

```
begin integer i; real array x[1:40]; M: NLCR;
   for i:=1, i+1 while i≤40 do x[i]:= read;
   for i:=1, i+1 while i≤40 do
      begin NLCR; print(i); TAB; print(x[i]) end;
   goto M end
```

---

t-arc hive.org

TEST 22

```
begin real x; integer y; Boolean B;
x:= 2; y:= 3; B:= true; L:
NLCR; print ( if B. then x else y ); B:= ¬B; goto L end
```

TEST 23

```
begin L: NLCR; print (XEEN(-o)); goto L end
```

---

t-larchive.org

```
begin  real x, y, z, X2, t, T;
       integer n, k;

for n:= 0 step 5 until 125 do

  begin NLCR; print(n); TAB;

    X:= n/360 * 3.14159 26535 89793;

    X2:=-X * X; T:=y:= z:= t:= X;

    for k:= 1, k+2 while true do

      begin t:= t * X2/(k+1)/(k+2); y:= y + t;

        T:= X2/(k+2) * T/(k+1); z:= z+ T;

          if y-t = y then goto SIN end;

  SIN: print(y); TAB; print(z) end end
```

t-larchive.org

```
comment TEST 25;
begin integer i,j;
    integer array a[1:5,0:4];
    real array b[1:5,0:4];
    for i:=1 step 1 until 5 do
      for j:=0 step 1 until 4 do
        begin a[i,j]:=i*(j+10); b[i,j]:=(i+10)*j end;
    begin
      integer array c[1:5,0:4.2];
      array d[1.3:5,0:4];
      procedure DRUK(u); value u; integer array u;
      begin NLCR; for i:=1 step 1 until 5 do
                  for j:=0 step 1 until 4.0 do
                  begin NLCR; print(u[i,j]) end
      end DRUK;
      for i:=1 step 1 until 5 do
        for j:=0 step 1 until 4 do
          begin c[i,j]:=i*(j-10); d[i,j]:=(i-10)*j end
L:    DRUK(a); DRUK(b); DRUK(c); DRUK(d);
      goto L end end
```

tlcarchive.org

Test 27.
Test 28.

```
comment TEST 27;
begin integer i, j, k;
  integer array a[1:2, 1:2, 1:2]; array b[1:2, 1:2, 1:2];
  for i:= 1,2 do for j:= 1,2 do for k:= 1,2 do
    begin a[i, j, k] := 4 * i + 2 * j + k;
      b[i, j, k]:= i + 2*j + 4 * k end;
    begin integer array c[1:2, 1:2, 1:2]; array d[1:2, 1:2, 1:2
    procedure DRUK (u); value u; integer array u;
  begin NLCR;
  for i:= 1,2 do for j:= 1,2 do for k:= 1,2 do
  begin NLCR; print (u[i, j, k]) end
  end;
  for i:=1,2 do for j:= 1,2 do for k:= 1,2 do
    begin c[i, j, k] := 4*j + 2*i + k;
      d[i, j, k]:= 4*j + 2* k + i end;
L: DRUK (a); DRUK (b); DRUK (c); DRUK (d); goto L
end end
```

t-larchive.org

TEST 29
30
31

```
comment TEST 29 30;
begin integer i,j,k,l;
  integer array a[1:2,1:2,1:2,1:2];
  real array b[1:2,1:2,1:2,1:2];
for i:=1,2 do for j:=1,2 do for k:=1,2 do for l:=1,2
do begin a[i,j,k,l]:= 8*i+4*j+2*k+l;
      b[i,j,k,l]:= i+2*j+4*k+8*l end;
begin integer array c[1:2,1:2,1:2,1:2];
      array d[1:2,1:2,1:2,1:2];
 procedure DRUK(u); value u; (integer) array u;
 begin NLCR; for i:=1,2 do for j:=1,2 do for k:=1,2 do
  for l:=1,2 do begin NLCR; print(u[i,j,k,l]) end
 end;
 for i:=1,2 do for j:=1,2 do for k:=1,2 do for
  l:=1,2 do begin c[i,j,k,l]:= 8*j+4*k+2*i+l;
                  d[i,j,k,l]:= j+2*k+4*i+8*l
  end;
L: DRUK(a); DRUK(b); DRUK(c); DRUK(d); goto L
end end
```

rlcarchive.org

TEST 34
TEST 35

```
comment TEST 34;
begin real X;
  real procedure Ch.(n, x); value n, x; integer n; real x;
  begin real a, b, c; integer i;
    a:=1; b:=x; if n=0 then c:=a else if n=1 then
    c:=b else
      for i:=2 step 1 until n do
        begin c:=2 * x * b - a; a:=b; b:=c end;
    Ch:=c end;
  X:=0
TERUG: NLCR; print(entier(1000 * X + .5)); TAB;
  print(Ch(10, X * .5) * 2); X:=X+0.001; goto TERUG
end
```

minus

t-larchive.org

TEST 32
TEST 33.
vervijderd.

```
comment TEST 32 = Ch(10, X);
begin
real X;
real procedure Ch(n, x); value n, x; integer n; real x;
Ch:= if n<2 then (if n=0 then 1 else x) else
    Ch(n-1, x) * 2 * x - Ch(n-2, x);
X:=0;
TERUG: NLCR; print(entier(1000 * X + .5)); TAB;
print(Ch(10, X * .5) * 2); X:= X + 0.001; goto TERUG
end
```

TEST 32 3 vogels lq''

ttcarchive.org

TEST 37

```
begin
    switch S := S[2], D[1], L;
    switch D := D1 , S[1];
L: NLCR; goto D[2]; print (0); D1: print (1); goto S[3]
end
```

TEST 38

```
begin integer i, s;
    integer array a, b [1:5], c [1:5, 1:2];
    s := 0;
    for i := 1 step 1 until 5 do
    begin a[i] := b[i] := 2 * i - 1;
        c[i,1] := a[i]; c[i,2] := - b[i];
        s := s + c[i,1] - c[i,2] end;
NLCR; print (s)
end
```

t-larchive.org

test ALGOL 60 #1: 13-5-60

```
begin integer a,b,c; array d[c:100]; procedure PRESCAN; LAB:;
    for a:=1 step 1 until d[b,c] do KLAAS: for b:=7 do
    Kareltje: begin switch V:= Kareltje; a:=b; Jan; Piet: end end
```

DAONNODN

0	begin	+104	;	+91	j	+19	
1	integer	+108	for	+85	e	+14	
2	a	+10	a	+10	:	+90	
3	,	+87	:=	+92	begin	+104	
4	b	+11	1	+1	switch	+111	
5	,	+87	step	+99	V	+58	
6	c	+12	1	+1	:=	+92	
7	;	+91	until	+95	K	+97	
8	array	+110	d	+13	a	+10	
9	d	+13	[	+100	n	+27	
10	[	+100	b	+11	e	+14	
11	c	+12	,	+87	,	+87	
12	:	+90	c	+12	l	+21	
13	1	+1	]	+101	t	+29	
14	0	+0	do	+86	;	+19	
15	0	+0	K	+97	e	+14	
16	]	+101	l	+21	s	+91	
17	;	+91	a	+10	a	+10	
18	integer	+108	a	+10	:=	+92	
19	procedure	+112	s	+28	b	+11	
20	P	+52	:	+90	;	+91	
21	R	+54	for	+85	]	+46	
22	E	+41	b	+11	a	+10	
23	S	+55	:=	+92	n	+23	
24	C	+39	7	+7	:	+90	
25	A	+37	do	+86	P	+52	
26	N	+50	K	+97	i	+18	
27	;	+91	a	+10	e	+14	
28	L	+48	n	+27	t	+29	
29	a	+10	e	+14	:	+90	
30	b	+11	l	+21	end	+105	
31	:	+90	t	+29	end	+105	

TLI

PLIP: 3-31-11

12	15
13 / 14	PRESCAN
15	17
16	LAB
17	18
18	21
19 / 20	KLAAS
21	22
22	25
23 / 24	Kareltje
25	26
26	30
27	PIET
28	Jan
29	V
30	31
31	30

tricarchive.org

```
begin real X; integer A

procedure BISEC ( y1, y2, eps, eps1, f, x, FLSXT );
  value eps, eps1, y1, y2;
  real eps, eps1, y1, y2, f, x;
  label FLSXT;
  begin integer i, j, k; real f, f1;
        switch g := FV, SV; switch d := FLSXT, Rd; switch e := TV, Re;
    i := j := k := 1; x := y2; a: f := F; if abs(f) ≤ eps then
    goto RETURN; goto g[i];
FV: i := 2; f1 := f; x := y1; goto a;
SV: if sign(f) = sign(f1) then goto d[j]; goto e[k];
TV: j := k := 2;
MP: x := (y1 + y2)/2; goto a;
Rd: y2 := x;
PR: if abs(y1 - y2) ≥ eps1 then goto MP; goto RETURN;
Re: y1 := x; goto PR;
RETURN: end BISEC;

  A := 1; BISEC ( 0, A, 10-10, 10-10, X*X-A, X, L );
  NLCR; print(A); TAB; print(X);
  A := A+1; goto T; L: stop end
```

Test PRESCAN. 13-5-60

Adder RNS

```
     DA  0  Z Y0 D1
0    2B  6  Z Y0
1    2S  0    X0 B
2    6S  9  Z E0
3    0B  1       A
4    6B  6  Z X0
5    2T  8    X0    E =>
6    ZX1
=> 7 2S  0  N N0 A
8    6S  6  Z Y0
9    6T  0  Z Y0 0   => plaats begin
10   2T  0  H X0 A   =>
```

Voorspanning

```
DPZE   0 X  9
DPZK   0 ZE 2
DPZT   0 ZK 1
DPZY   0 ZT 1
DPFT   0 ZY 1
DPHF   0 FT 5
DPHH   0 HF 2
DPHK   0 HH 1
DPHL   0 HK 4
DPNN   0 HL 1

  DA  6ZE1 D1
  0A  4095 X0     PLIE
  DA  8ZE1
  0A  0NN10       TLIB
```

NB verspeten was 12ZE0 = FLIB intervallen!

FTL test herrop!

tlcarchive.org

```
comment TEST 40;
begin integer n,k,h;
    real procedure SIGMA(i,j,k,t); value j; integer i,j,k; real t;
    begin if j>k then SIGMA:=0 else
        begin i:=j; SIGMA := t + SIGMA(i,j+1,k,t) end
    end;
NLCR; print(SIGMA(n,1,10,SIGMA(k,1,n,2)-1));
NLCR; print(SIGMA(n,1,10,SIGMA(k,1,n,k-6)+1)-1); NLCR;
print(SIGMA(n,1,10,SIGMA(k,1,n,SIGMA(h,1,k,6)-6)+1)-1)
end
```

t-larchive.org

```
comment
TEST 50   EVDJA2 101060;
  begin integer n, k, h;
  NLCR; print (SUM (n, 1, 10, SUM (k, 1, n, 2) - 1));
  NLCR; print (SUM (n, 1, 10, SUM (k, 1, n, 6 * k - 6) + 1) - 1);
  NLCR; print (SUM (n, 1, 10, SUM (k, 1, n, SUM (h, 1, k, 6) - 6) + 1));
  NLCR; print (SUM (n, 1, 1000, n * n));
  NLCR; print (SUM (n, 1, 10000, n * n));
  for k := 1 step 1 until 10000 do
begin stop; NLCR; for h := 1 step 1 until 5 do print (read)
end
end
```

t-larchive.org

```
integer procedure FIBONACCI (fi); boolean fi;
begin if fi then begin
begin own integer n;
    if fi then n:=0 else n:= n+1;
    begin own array A[n : n+2];
        if fi then begin A[n]:=0; A[n+1]:= 1; end
        FIBONACCI := A[n+2]:= A[n]+ A[n+1]
    end
end
```

A[0] =

0
1
1
2
3
5
8
13
21
.
.
.

t-1carchive.org