K O N I N K L IJ K  N E D E R L A N D S
M E T E O R O L O G I S C H  I N S T I T U U T

D e  B i l t

Verslagen
V - 231

# SPECIALE PROJEKTGROEP NUMERIEKE VOORSPELMETHODEN

## Een experiment met codeprocedures

dr. D.J. Bouman

De Bilt, 1971

Kon. Ned. Meteor. Inst.
De Bilt

Publikatienummer: KNMI V-231 (II)

U.D.C.: 681.14

KONINKLIJK NEDERLANDS
METEOROLOGISCH INSTITUUT

Verslagen V-231-II

SPECIALE PROJEKTGROEP NUMERIEKE VOORSPELMETHODEN

# Een experiment met codeprocedures

door

dr. D.J. Bouman

## 1. Inleiding

In Verslagen V-220 (1969) werd een voorlopige handleiding voor het samenstellen van codeprocedures gegeven. Een druk gebruik van codeprocedures is sindsdien door niemand gemaakt, ook niet door de auteur van de handleiding. De redenen liggen nogal voor de hand. Het schrijven van een codeprocedure is een tijdrovende bezigheid en de kansen, dat moeilijk op te sporen fouten worden gemaakt, zijn niet te onderschatten. In V-220 werd reeds gesteld, dat het aantrekkelijk zou zijn als de codeprocedures in ELAN zouden kunnen worden geschreven en na een precompilatie in het ALGOL-programma ingelast. Inmiddels is gebleken, dat er inderdaad mogelijkheden bestaan deze minder tijdrovende en minder riskante weg te volgen en hierover handelt dit geschrift.

## 2. Het produktieprogramma PCP

De ontdekking van bovengenoemde mogelijkheden was het gevolg van een wenk van een employé van Electrologica, welke firma een ELAN-programma PCP (Produce Code Procedures) bleek te hebben ontworpen, waarmede in ELAN geschreven subroutines geprecompileerd kunnen worden tot codeprocedures, die zonder meer in een ALGOL-programma kunnen worden ingelast. Een ponsband van PCP is sinds de oplevering van de X8 in het bezit van het KNMI geweest.

Het bleek, dat de band dateerde van voor de tijd dat de X8 met een regeldrukker was uitgerust maar aanpassing aan de huidige situatie was niet moeilijk. Na enkele proefjes werd besloten tot een experiment op grotere schaal.

## 3. Beschrijving van het experiment

Besloten werd voor een eerste experiment een zestal procedures te kiezen, die voorkomen in:

T.J. DEKKER: ALGOL-60 Procedures in Numerical Algebra, Part 1, M.C. A'dam. MC-Tracts no. 22, 1963 (procedures 2000 t/m 2005, l.c. blz. 8).

Het gaat hier om procedures, die de berekening van inwendige produkten van vektoren en matrices bewerkstelligen en die door DEKKER zijn ontworpen om dienst te doen in zijn pakket van snelle procedures voor matrixinversie, determinantberekening, eigenwaardeproblemen en oplossing van stelsels lineaire vergelijkingen. Bovengenoemde zes procedures zijn door het M.C. A'dam en ERCU-Utrecht inmiddels in hun bedrijfssysteem opgenomen.

Sinds enige tijd was de schrijver in het bezit van de ELAN-tekst van deze procedures, in fotokopie verkregen van het M.C. door bemiddeling van prof. Kruseman Aretz. Het enorme werk, verbonden aan het met potlood en papier omzetten van een ELAN-tekst van vrij grote lengte in een codeprocedure, was er de oorzaak van dat dit tot nu toe niet was gebeurd.

Er deed zich dus een goede gelegenheid voor het via het PCP-programma te doen. Wel moest een aantal veranderingen in de tekst, zoals die door het MC was gemaakt, worden aangebracht, als gevolg van de omstandigheid dat de beschikbare tekst bestemd was om het bedrijfssysteem uit te breiden en niet om als codeprocedure te dienen. In paragraaf 5 zal hierover meer worden medegedeeld.

De in het experiment gebruikte variatie op de cantus firmus, alsmede de door PCP hieruit geproduceerde codeprocedure "dekker" zijn opgenomen in een testprogramma "MATS", dat in zijn geheel als bijlage 1 is gereproduceerd. MATS bevat voldoende commentaar, zodat voor de werking van "dekker" hiernaar kan worden verwezen. In MATS zijn alle codeprocedures en enige hiermee corresponderende ALGOL-procedures opgenomen, zodat een vergelijking in MATS zelf kon worden uitgevoerd.

## 4. Resultaten van het experiment

Zoals uit de als bijlage 2 gereproduceerde output van het proefprogramma MATS blijkt, werken de codeprocedures naar wens. De verhouding van de snelheden der procedures in ALGOL en in code blijkt in het onderzochte geval 5.24 te zijn. Uit de gegevens van DEKKER volgt, dat hij in eenzelfde situatie ongeveer 5.75 gevonden heeft. De omstandigheden zijn in zijn geval iets gunstiger dan in het beschreven experiment (procedures ingebouwd in het bedrijfssysteem contra codeprocedures), zodat het kleine verschil van 5.24 tegen 5.75 eerder mee- dan tegenvalt. Volgens DEKKER neemt de verhouding der snelheden toe met het groter worden der vektoren en matrices en nadert asymptotisch tot 7.4.

Ook de foutmelding werkt korrekt. Merk op dat het regelnummer, dat in de foutmelding wordt vermeld (419), het nummer is van de regel van de ALGOL-tekst waarin de aanroep van de codeprocedure voorkomt. (Zie ook para 5.7).

### 4.1 Conclusies uit het experiment

Het geheel levert het bewijs, dat zelfs tamelijk omvangrijke snelle codeprocedures op betrekkelijk gemakkelijke wijze via PCP kunnen worden vervaardigd. Het tijdrovende en riskante werk, dat verbonden is aan het met potlood en papier maken van een codeprocedure, kan geheel worden vermeden. Het is een mooi voorbeeld van automatisering.

Zoals reeds in V-220 werd uiteengezet, zijn snelle codeprocedures van groot belang voor langdurende routineprogramma's, die veelvuldig te doorlopen cycli bevatten. Het beschikbaar komen van PCP biedt de mogelijkheid meer gebruik te maken van codeprocedures dan tot nu toe gebeurde.

## 5. Technische details*

5.1 Om het PCP-programma te kunnen gebruiken, moet men van de hieronder te beschrijven details op de hoogte zijn.

PCP is geen volledig ELAN-programma, maar slechts het eerste deel van een programma. Het tweede deel moet de gebruiker zelf maken en dit moet de ELAN-tekst van de te produceren codeprocedure bevatten volgens onderstaande voorschriften.

---

* Kan door de niet-geïnteresseerde lezer zonder bezwaar worden overgeslagen.

5.2 Na facultatief ELAN-commentaar begint het tweede deel van PCP met

FIRST: 'BEGIN' 'MT'

De label FIRST is verplicht. Na 'MT' volgen alle lokale namen (labels, inzetaanwijzingen), die in de te precompileren ELAN-tekst voorkomen.

5.3 Daarna volgt de eigenlijke tekst. Deze bestaat uit gewone ELAN-opdrachten en ELAN-MACRO'S. De ELAN-MACRO'S staan in (1,1) correspondentie met de macro's van de ALGOL-compiler zoals zij zijn beschreven in V-220 (N.B. : de twee soorten macro's zullen door verschil in lettertype in het vervolg worden onderscheiden).

5.4 De laatste te precompileren opdracht moet worden voorzien van de globale label LAST. Het slot van het tweede deel ziet er dan als volgt uit:

```
LAST: '+++++++'
      'END'
'START' :START
'END'
'END'
'END'
'END'
```

(N.B.: LAST mag dus niet voorkomen in de lijst van lokale labels achter 'MT').

5.5 Zoals uit 5.2 blijkt, heerst in het tweede deel van PCP de symbolische MT-adressering. De konsekwenties hiervan moeten in acht worden genomen. Deze zijn:

5.5.1 Verboden zijn:
Statische adressering (behalve in vormen als A+1 c.q. A+:M[1]),
B-modificatie bijv. JAN [B+3],
MD-adressering,
Mp[q]-adressering.

Een beperking houden deze verboden niet in, daar voor met potlood en papier gemaakte codeprocedures dezelfde verboden gelden.

5.5.2 Het is onmogelijk opdrachten te gebruiken waarin statische adressering vereist is. Dit zijn:
SUB; REP; REPP; REPZ; REPE;

Het ontbreken van SUB is geen bezwaar. Hiervoor kan men in de praktijk altijd SUBC gebruiken. Het ontbreken van het viertal repeterende sprongen is een klein ongemak, dat echter ook wordt ondervonden in met potlood en papier gemaakte codeprocedures.

5.5.3 De bovengenoemde verboden gelden niet voor procedures opgenomen in het bedrijfssysteem. De verboden constructies kwamen dan ook in de tekst van het M.C. voor de Dekker-procedures regelmatig voor. Voor gebruik als codeprocedures moesten zij door andere opdrachten worden vervangen, hetgeen meestal een beperkte reorganisatie van het programma eiste.

5.5.4 Zoals bekend, moet bij MT-adressering de parameter q in de geheugenoperand MT[q] voldoen aan

$$-256 \leq q \leq 255$$

De omvang van de procedure is daardoor aan de beperking gebonden, dat geen MT-adresseringen mogen voorkomen, die niet aan deze voorwaarde voldoen. Wat dit betreft is de procedure **dekker** met haar 238 opdrachten aan de grens van de mogelijkheden. Op de vraag hoe te handelen als de procedure te groot gaat worden, wordt onder 6.1 ingegaan.

5.6 Het gebruik van systeemadressen is niet verboden, maar maakt de codeprocedures op gevaarlijke wijze systeemafhankelijk.

5.7 Beschikbare macro's

De ELAN-MACRO'S, die in het eerste deel van PCP worden gedefinieerd en in het tweede deel kunnen worden gebruikt, worden met dezelfde namen aangeroepen als waarmede zij in de ALGOL-compiler symbolisch worden aangeduid. Opgenomen zijn:

IDI, TTP, TSR, TSI, TSB, TFSU, TSL, TFSL, TCST, STSR, STSI, SSTSI, STSB, STFSU, CRV, CIV, CBV, CLV, CEN, CLPN, ENTIER, SQRT, EXP, LN, SIN, COS, ARCTAN , PEND, PREAD, PREHEP.

Op het hieronder te bespreken viertal na hebben zij de betekenis als aangegeven is in V-220. De bewerking, die PCP moest ondergaan om uitvoer via de regeldrukker als extraatje bij de uitvoer van de ponser mogelijk te maken, werd benut om de volgende vier macro's toe te voegen:

TCST : Foutmelding met foutnummer 600.
In MATS vindt men voorbeelden van de mogelijkheid deze macro ook voor foutmeldingen met andere foutnummers te gebruiken. Daar de (statische) systeemadressen ERRORTABLE en ERRORM in codeprocedures niet beschikbaar zijn, moesten de foutmeldingen, zoals die in de oorspronkelijke tekst van het M.C. voorkwamen, worden gewijzigd. Zoals in MATS te zien is, kon hiervoor een eenvoudige oplossing worden gevonden, die ook in andere procedures kan worden toegepast.

PEND : beëindiging ALGOL-programma.

PREAD : read in ALGOL-programma.

PREHEP: REHEP in ALGOL-programma.

Een groot aantal macro's dat in V-220 werd beschreven, vindt men niet terug in PCP. De ontbrekende kunnen worden onderscheiden in twee soorten:

(a) de overbodigen. Typisch voorbeeld hiervan is de macro met nummer 0, die in het ALGOL-programma de opdracht **MC=F** genereert. Deze is nu overbodig geworden, omdat men in de ELAN-tekst zonder meer **MC=F** als opdracht kan schrijven;

(b) de macro's met parameter. Voor zover het macro's betreft, die als parameter een getal moeten meekrijgen, behoren zij eigenlijk in categorie (a) thuis. Het ontbreken van MACRO'S, die corresponderen met met macro's waaraan een ALGOL-identifier als parameter moet worden meegegeven, is een ongemak. Het heeft tot konsekwentie dat de communicatie van de codeprocedure met het omhullende ALGOL-programma geheel moet verlopen via de formele parameters van de procedure. De selectie van wel opgenomen MACRO'S garandeert dat in de CP geen macro's voorkomen, die de ALGOL-compiler zouden kunnen verleiden tot het toepassen van een optimalisatie, hetgeen de MT-adressering danig in de war zou sturen.

## 5.8 Voorbeeld

Daar de toepassing in MATS nogal gecompliceerd is, zal hier een eenvoudiger voorbeeld worden gegeven van hoe het tweede deel van PCP er uit kan zien.

```
FIRST:  'BEGIN'  'MT'  AGAIN
                S=0                 "STANDAARD INGANG
                A=B                 "PARAMETER WIJZER
                CIV                 "SYSTEEM MACRO CALL INTEGER
                                    "PARAMETER BY VALUE
               A=MC[-1]             "PARAMETER
                F=1                 "INITIALISATIE
              U,A-1,P               "IS PARAMETER >1?
            N,GOTOR(MC[-1])         "ZO NIET DAN KLAAR
AGAIN:          G=A                 "REGISTER = PARAMETER
               A-1,P                "LAAG PARAMETER AF
                                    "IS PARAMETER NU NOG > 0?
             Y,GOTO(:AGAIN)         "JA, DAN TERUG NAAR AGAIN
LAST :       GOTOR(MC[-1])          "KLAAR EN TERUG PROGRAMMA
                'END'
'START'         :START
'END'
'END'
'END'
'END'
```

Men herkent hierin gemakkelijk een procedure voor faculteit (integer value parameter). Merk op dat de eigenlijke cyclus slechts drie opdrachten telt en dus zeer snel is.

Na precompilatie via PCP verkrijgt men dan de volgende code-procedure:

```
real procedure fac (n); value n; integer n;
< 128, 21495808, 128, 4194315, 54, 128, 579 8655, 128, -288 35838,
  128, 278 5281, 128, -44303616, 128, -1674488, 128, 275 2513,
  128, -42 369794, 128, -44401920 > ;
```

Analyseert men de voorlaatste opdracht

```
128, -42 369794,
```

dan blijkt dat PCP de opdracht

```
Y,GOTO(:AGAIN)
```

geïnterpreteerd heeft als

```
Y,GOTO(:MT[-3])
```

precies zo, als men van een assembler, die onder MT-adresseringswijze werkt, mag verwachten.

## 6. Het uitgebreide produktieprogramma PCPU

Het enige gebrek, dat men ELECTROLOGICA's PCP kan verwijten, is dat de communicatie van de codeprocedures met de omhullende ALGOL-programma's geheel moet verlopen via de formele parameters van de procedures. Dit is een gevolg van het ontbreken in PCP van ELAN-MACRO'S, die corresponderen met ALGOL-compiler macro's waaraan een ALGOL-identifier als parameter moet worden meegegeven. Daarom is een uitbreiding PCPU ontworpen, die hieraan tegemoet komt, waardoor de beschikking is verkregen over de navolgende macro's, voor de werking waarvan wordt verwezen naar V-220:

TRV, TIV, TBV, TAK, STR, STI, SSTI, STB, JU1, IJU, IJU1, SUBJ, ISUBJ.

Hiermede wordt een soepeler communicatie tussen codeprocedure en programma mogelijk. Men bedenke echter, dat het voorkomen van dit type macro's in een codeprocedure haar programma-afhankelijk kan maken, zodat codeprocedures waarin van de nieuwe mogelijkheden gebruik wordt gemaakt, niet zonder voorzorgen van het ene programma naar het andere kunnen worden getransplanteerd. Ook de toegevoegde macro's kunnen de ALGOL-compiler niet tot optimalisatie verleiden.

### 6.1 Technische details

In de ALGOL-teksten wordt vrijelijk gebruikgemaakt van hoofdletters en kleine letters. Hoewel de regeldrukker slechts met één lettertype is uitgerust, is hierin voorzien doordat met een merkteken door de letter heen hoofdletters van kleine letters worden onderscheiden. In ELAN pleegt men van slechts één lettertype gebruik te maken, nl. hoofdletters, en in de output-procedures voor ELAN-programma's op de regeldrukker kan dan ook maar van één lettertype worden gebruikgemaakt. In de uitvoer van ELAN-programma's via de ponser kunnen weer beide types worden gebruikt. Voor PCPU houdt dit in, dat in de output via de regeldrukker kleine letters en hoofdletters worden vereenzelvigd, maar dat in de output via de ponser onderscheid wordt gemaakt. Daar de uitvoer via de regeldrukker slechts een extraatje is, vormt dit geen bezwaar. (In het ALGOL-programma wordt de ponsband ingelast!).

Het begin van de tweede band van PCPU moet er nu enigszins anders uitzien dan in para. 5 werd aangegeven. De te volgen methode kan het beste met behulp van een voorbeeld worden beschreven. Stel, in het ALGOL-programma komen als identifiers waaraan men in een codeprocedure wil refereren, voor:

BACH, BEETHOVEN, Brahms, Bruckner, BARTOK.

De band moet dan aldus beginnen:

```
BACH      : '('BACH')'        "KK=4
BEETHOVEN: '('BEETHOVEN')'  "KK=9
BRAHMS    : '('Brahms')'      "KK=6
BRUCKNER : '('Bruckner')'    "KK=8
BARTOK    : '('BARTOK')'      "KK=6
FIRST     : 'BEGIN' 'NT'      "ENZ.
```

De alfanumerieke informatie tussen '(' en ')' moet gegeven worden in dezelfde lettertypen als gebruikt in de ALGOL-tekst. De hiervoor staande namen (inzetaanwijzingen) moeten geheel in hoofdletters worden geschreven. Men is vrij in de keuze van de namen van de inzetaanwijzingen maar het zal dikwijls mogelijk zijn hiervoor namen te kiezen, die min of meer overeenkomen met de ALGOL-identifiers. Als geheugensteuntje is achter elke alfanumerieke informatie in het commentaar het aantal symbolen van de informatie gegeven. (Zoals bekend neemt een informatie van KK-symbolen

(KK+2)÷3 geheugenplaatsen in beslag, waar de gebruiker overigens niets mee te maken heeft). Voor KK wordt echter in PCPU geeist dat voldaan is aan $1 \leq KK \leq 10$.

Wil men nu op een bepaalde plaats in een codeprocedure de compiler-macro

92, Bruckner

genereren, dan moet in PCPU op de overeenkomstige plaats tussen FIRST en LAST staan

TAK (BRUCKNER,8)

De eerste parameter van de MACRO TAK is de inzetaanwijzing, de tweede geeft het aantal symbolen van de ALGOL-identifier.

Enige proefjes hebben bewezen, dat PCPU een waardevolle uitbreiding van PCP is, die vooral van belang kan zijn voor programma's met veel variabelen, die moeilijk in groten getale aan een codeprocedure via het formeel-actueel mechanisme kunnen worden meegegeven.

PCPU kan verder goede diensten bewijzen als de procedure te groot dreigt te worden (zie 5.5.4). Zij kan nu in delen gesplitst worden, die elkaar via SUBJ, JU1, enz. kunnen aanroepen en die verder communicatie met elkaar kunnen onderhouden zowel via ALGOL-identifiers als via formele parameters.

## 7. Bijlagen en Slotopmerkingen

Bijlage 1 : Volledige tekst van het proefprogramma MATS, inclusief de ELAN-tekst van de codeprocedure "dekker".

Bijlage 2 : Output van MATS.

Bijlage 3 : Tekst van PCPU. De uitbreidingen t.o.v. PCP zijn duidelijk als zodanig aangegeven.

Opmerkingen:

1. Belanghebbenden kunnen bij de auteur kopieën van de verschillende banden verkrijgen. Zodra ze na verloop van tijd hun definitieve vorm hebben, zullen ook exemplaren in de computerzaal worden gedeponeerd.

2. Hoewel op een paar typefoutjes na V-220 korrekt is, zijn grote delen hiervan door het beschikbaar komen van PCP (c.q. PCPU) niet meer van belang. Met name kan een aantal nogal omslachtige constructies drastisch worden vereenvoudigd. V-220 blijft echter van betekenis, voor zover daarin de struktuur van het objektprogramma wordt beschreven, iets waarmede schrijvers van codeprocedures goed op de hoogte moeten zijn.

3. Tijdens de samenstelling van dit verslag kwam ter kennis van de schrijver:

   K.K. KOKSMA: The ELAN Source text of MICRO 64 KL,
   M.C. A'DAM MR 116/70, 1970.

   Hierin is de tekst (in ELAN) van de procedures matmat enz. gepubliceerd. De nieuwe tekst wijkt enigszins af van de tekst waarop het experiment is gebaseerd, omdat daarin rekening wordt gehouden met een geheugen van 64 K, waarvan de adressen met nummer groter dan 32767 niet zonder extra bewerkingen kunnen worden bereikt. Voor het overige zijn geen wijzigingen geconstateerd.

De Bilt, dec.'70-jan.'71.

# I N H O U D



-c-o-o-

```
begin   comment BOUD 241270 MATS.
        Proefprogramma voor codeprocedures matmat enz.;

        integer i,j,k;
        real x,y,z,t;
        integer array v[1:50],p[1:40,1:50];
        array q[1:40,1:50];

        real procedure algolvecvec(l,u,shift,a,b); value l,u,shift;
                integer l,u,shift; array a,b;
        begin   integer i; real s; s:=0;
                for i:= l step 1 until u do s:=a[i]×b[shift+i]+s;
                algolvecvec:=s
        end algolvecvec;

        real procedure algolmattam(l,u,i,j,a,b); value l,u,i,j;
                integer l,u,i,j; array a,b;
        begin   integer k; real s;
                s:=0;
                for k:= l step 1 until u do s:=a[i,k]×b[j,k]+s;
                algolmattam:=s
        end algolmattam;

        comment bovenstaande procedures zijn overgeschreven uit het boek van DEKKER
                en dienen alleen om de resultaten te kunnen vergelijken met die
                verkregen d.m.v. de corresponderende codeprocedures;
        comment in the following comments array a[l:u] etc. means the segment a[i],
                l≤i≤u of the array a[L:U], L≤l≤u≤U;

        comment vecvec:=scalar product of the vectors given in array a[l:u] and array
                b[shift+l:shift+u];
        real procedure vecvec(l,u,shift,a,b); value l,u,shift;
                integer l,u,shift; array a,b;
                ⊰121,0,87,1,103,dekker⊱;

        comment matvec:= s.p. of the rowvector array a[i:i,l:u] and the vector b[l:u];
        real procedure matvec(l,u,i,a,b); value l,u,i; integer l,u,i;
                array a,b;
                ⊰121,0,87,2,103,dekker⊱;

        comment tamvec:= s.p. of the column vector array a[l:u,i:i] and the vector b[l:u];
        real procedure tamvec(l,u,i,a,b); value l,u,i; integer l,u,i; array a,b;
                ⊰121,0,87,3,103,dekker⊱;

        comment matmat:= s.p. of the rowvector array a[i:i,l:u]
                and the column vector array b[l:u,j:j];
        real procedure matmat(l,u,i,j,a,b); value l,u,i,j; integer l,u,i,j; array a,b;
                ⊰121,0,87,4,103,dekker⊱;

        comment tammat:= s.p. of the column vectors array a[l:u,i:i] and array b[l:u,j:j];
        real procedure tammat(l,u,i,j,a,b); value l,u,i,j; integer l,u,i,j; array a,b;
                ⊰121,0,87,5,103,dekker⊱;

        comment mattam:= scalar product of the row vectors
                array a[i:i,l:u] and array b[j:j,l:u];
        real procedure mattam(l,u,i,j,a,b); value l,u,i,j; integer l,u,i,j; array a,b;
                ⊰121,0,87,6,103,dekker⊱;
```

```
        comment de codeprocedures hierboven gegeven doen niets anders dan aan vecvec,matvec,tamvec,matmat,tammat
                en mattam de nummers 1 t.e.m. 6 toekennen en via een sprongopdracht de procedure dekker ingaan;

        comment het is niet nodig de procedures die men niet gebruikt in de
                ALGOL tekst op te nemen. Wel altijd dekker;

        comment nu volgt de ELAN tekst van de procedure dekker:
                                ''COMMENT IN ENGELS OORSPRONKELIJK,IN NEDERLANDS TOEGEVOEGD
                                ''procedure dekker,vrij naar MC-ADAM
                                '' BOUD dec.1970.

FIRST:          'BEGIN' 'MT' MATMAT,BKJ,MATMAT8,MATMAT9,TAMMAT,
                MATTAM,BJK,MATVEC,BK,TAMVEC,VECVEC,PLIST22,
                PLIST222,PLIST223,PLIST21,PLIST11,PLIST2,PLIST25,
                PLIST1,TESTDIM,IND1,IND2,AIK,AKI,INPR,RR,RI,AI,
                IR,II,DA,DB,SUM,DOWN,MASK,FOUT

                S=0             ''STANDAARD INGANG
                JUMP(G)         ''DISTRIBUTIE SPRONG
MASK:           TCST            ''KOMT NIET VOOR: FOUTMELDING=600
                GOTO(:VECVEC)
                GOTO(:MATVEC)
                GOTO(:TAMVEC)
                GOTO(:MATMAT)
                GOTO(:TAMMAT)
                GOTO(:MATTAM)

FOUT:           S=MASK          ''TOEVOEGING: FOUTMELDING
                S'x'32767       ''MET FOUTNUMMER
                GOTO(:MS[1])    ''MEEGEGEVEN IN A

MATMAT:         SUBC(:PLIST22)
                SUBC(:AIK)      ''adress a[i,1] and [i,u]
BKJ:            A=M[B-4]        ''b
                S=M[B-10]       ''1
                G=M[B-7]        ''j
                SUBC(:IND2)     ''adress b[1,j]
                S=M[B-10]       ''u
                G=M[B-8]        ''j
MATMAT8:        SUBC(:IND2)     ''adress b[u,j]
MATMAT9:        SUBC(:INPR)     ''inner product
                B-14
                GOTOR(MC[-1])

TAMMAT:         SUBC(:PLIST22)
                SUBC(:AKI)      ''adress a[1,i] and a[u,i]
                GOTO(:BKJ)

MATTAM:         SUBC(:PLIST22)
                SUBC(:AIK)      ''adress a[i,1] and a[i,u]
BJK:            A=M[B-4]        ''b
                S=M[B-7]        ''j
                G=M[B-10]       ''1
                SUBC(:IND2)     ''adress b[j,1]
                S=M[B-8]        ''j
                G=M[B-10]       ''u
                GOTO(:MATMAT8)
```

```
MATVEC:     SUBC(:PLIST21)
            SUBC(:AIK)      ''adress a[i,1] and a[i,u]
BK:         A=M[B-4]        ''b
            S=M[B-10]       ''l
            SUBC(:IND1)     ''adress b[l]
            S=M[B-10]       ''u
            SUBC(:IND1)     ''adress b[u]
            GOTO(:MATMAT9)

TAMVEC:     SUBC(:PLIST21)
            SUBC(:AKI)      ''adress a[1,i] and a[u,i]
            GOTO(:BK)

VECVEC:     SUBC(:PLIST11)
            A=M[B-4]        ''a
            S=M[B-8]        ''l
            SUBC(:IND1)     ''adress a[l]
            S=M[B-8]        ''u
            SUBC(:IND1)     ''adress a[u]
            A=M[B-4]        ''b
            S=M[B-10]       ''l
            S+M[B-8]        ''shift
            SUBC(:IND1)     ''adress b[l+shift]
            S=M[B-10]       ''u
            S+M[B-9]        ''shift
            SUBC(:IND1)     ''adress b[u+shift]
            GOTO(:MATMAT9)

PLIST22:    SUBC(:PLIST2)
            S=2             ''dimension of b
PLIST222:   F=2             ''dimension of a
PLIST223:   SUBC(:TESTDIM)
            G=M[B-7]        ''u
            G-M[B-8]        ''l
            COUNT=G
            F+1,P           ''u-l+1>0?
            Y,GOTOR(M[B-10])        ''sum not empty
            F=0             ''empty sum
            B-10            '' return over program link
            GOTOR(MC[-1])

PLIST11:    SUBC(:PLIST1)
            S=1             ''dimension of b
            F=1             ''dimension of a
            GOTO(:PLIST223)

PLIST21:    SUBC(:PLIST1)
            S=1             ''dimension of b
            GOTO(:PLIST222)

PLIST2:     A=:MC[-2]       ''parameter pointer
            CIV             ''l
            CIV             ''u
            CIV             ''i
            CIV             ''j
PLIST25:    CEN             ''a
            CEN             ''b
            S=M[B-3]        ''APIC1 of a
            RUS(20)
```

```
                s-9,P
                M[B-3]=S        ''real=-1, integer=-0
                S+2,E           ''not real or integer array?
                N,DOS(M[B-4])   ''execute APIC0 of a
                N,M[B-4]=A      ''adress of ar0 of a
                S=M[B-1]        ''APIC1 of b
                RUS(20)
                N,S-9,P
                M[B-1]=S        ''real=-1, integer=-0
                N,S+2,E         ''not real or integer array?
                N,DOS(M[B-2])   ''execute APIC0 of b
                N,M[B-2]=A      ''adress of ar0 of b
                N,GOTOR(M[B-9])         ''exit PLIST2
                A=522           ''FOUTMELDING(522)
                GOTO(:FOUT)

PLIST1:         A=:MC[-2]       ''parameter pointer
                CIV             ''l
                CIV             ''u
                CIV             ''i or shift
                B+1
                GOTO(:PLIST25)

TESTDIM:        A=M[B-5]        ''a , OORSPRONKELYK ALS SUB
                G-MA[-2],Z      ''dimension of a correct?
                A=M[B-3]        ''b
                Y,S-MA[-2],Z    ''dimension of b correct?
                Y,GOTOR(MC[-1])
                A=523           ''FOUTMELDING(523)
                GOTO(:FOUT)

IND1:           'BEGIN' 'MT' RED        ''AFWIJKING ZIE PARA 5.7.
                RED=S           ''OORSPRONKELYK EEN PROCEDURE
                S=MC[-1]        ''DIE MET SUB WERD AANGEROEPEN
                LINK=S
                S=RED
                U,S-MA[-5],P
                S-MA[-4],E      ''index outside bounds?
                Y,A=502         ''FOUTMELDING(502)
                Y,GOTO(:FOUT)
                U,S=MA[-6],P    ''integer array?
                N,LUS(1)        ''factor 2 for real array
                S+MA            ''+adress of last element
                MC=S
                GOTOR(LINK)
RED:            +0
                'END' IND1

IND2:           'BEGIN' 'MT' RED        ''OORSPRONKELYK ALS SUB
                RED=S
                S=MC[-1]
                LINK=S
                S=RED
                G-MA[-5],P
                G+MA[-5]
                G-MA[-4],E      ''second index outside bounds?
                Y,A=502         ''FOUTMELDING(502)
                Y,GOTO(:FOUT)
```

```
                GXMA[-6]        ''(i1-u1)xdelta1
                U,S-MA[-8],P
                S-MA[-7],E      ''first index outside bounds?
                Y,A=503         ''FOUTMELDING(503)
                Y,GOTO(:FOUT)
                U,S=MA[-9],P    ''integer array?
                N,LUS(1)        ''factor 2 for real array
                G+MA            ''+adress of last element
                S+:MG
                MC=S
                GOTOR(LINK)
RED:            +0
                'END' IND2

AIK:            'BEGIN' 'MT' RED,RED1           ''OORSPRONKELYK ALS SUB1
                RED=S
                S=MC[-1]
                RED1=S
                S=RED
                A=M[B-4]        ''a
                S=M[B-6]        ''i
                G=M[B-8]        ''1
                SUBC(:IND2)     ''adress of a[i,1]
                S=M[B-7]        ''i
                G=M[B-8]        ''u
                SUBC(:IND2)     ''adress a[i,u]
                GOTOR(RED1)
RED:            +0
RED1:           +0
                'END' AIK

AKI:            'BEGIN' 'MT' RED,RED1           ''OORSPRONKELYK ALS SUB1
                RED=S
                S=MC[-1]
                RED1=S
                S=RED
                A=M[B-4]        ''a
                S=M[B-8]        ''1
                G=M[B-6]        ''i
                SUBC(:IND2)     ''adress a[1,i]
                S=M[B-8]        ''u
                G=M[B-7]        ''i
                SUBC(:IND2)     ''adress a[u,i]
                GOTOR(RED1)
RED:            +0
RED1:           +0
                'END' AKI

DA:             +0
DB:             +0
SUM:            +0
                +0
DOWN:           F+SUM           ''+sum, DOWN TOEGEVOEGD
                A+DA            ''increment array a
                S+DB            ''increment array b
                MC=S            ''save S
                S=1
                MINS(COUNT)     ''count down
                U,S+1,Z         ''all done?
```

```
                S=MC[-1]
                N,GOTOR(MC[-1])         ''terug naar INPR
                B-1
                GOTOR(MC[-1])   ''verdwijn uit INPR

INPR:           G=M[B-4]
                G-M[B-5]
                G/COUNT
                DA=G            ''step of a
                G=M[B-2]
                G-M[B-3]
                G/COUNT
                DB=G            ''step of b
                A=M[B-5]        ''first element of a
                S=M[B-3]        ''first element of b
                F=0             ''initial value
                U,A=M[B-8],Z    ''a an integer array?
                Y,GOTO(:AI)
                U,A=M[B-6],Z    ''b an integer array?
                Y,GOTO(:RI)
RR:             SUM=F           ''save sum
                F=MA
                F×MS            ''element × element
                SUBC(:DOWN)     ''AFWIJKING ZIE PARA 5.7. OORSPRONKELYK EEN REPE-OPDRACHT
                GOTO(:RR)
RI:             SUM=F
                F=MA
                G×MS
                SUBC(:DOWN)
                GOTO(:RI)
AI:             U,A=M[B-6],Z    ''b an integer array;
                Y,GOTO(:II)
IR:             SUM=F
                G=MA
                F×MS
                SUBC(:DOWN)
                GOTO(:IR)
II:             SUM=F
                G=MA
                G×MS
                SUBC(:DOWN)
LAST:           GOTO(:II)
                'END'
'START'         :START
'END'
'END'
'END'
'END'

                Hier eindigt de ELAN-tekst van dekker.
                Nu volgt de geprecompileerde (via PCP) tekst van de codeprocedure dekker;
        procedure dekker;

            128,21495808,128,-50331589,28,128,-42435284,128,-42435296,128,
            -42435289,128,-42435322,128,-42435311,128,-42435309,128,22576376,
            128,63471615,128,-42436350,128,-34964175,128,-34964085,128,5259260,
            128,22036470,128,-28262406,128,-34964110,128,22036470,128,-28262407,
            128,-34964113,128,-34964050,128,-64487409,128,-44401920,128,-34964187,
            128,-34964083,128,-42435340,128,-34964190,128,-34964100,128,5259260,
```

```
            128,22036473,128,-28262409,128,-34964125,128,22036472,128,-28262409,
            128,-42435343,128,-34964183,128,-34964109,128,5259260,128,22036470,
            128,-34964147,128,22036470,128,-34964149,128,-42435350,128,-34964191,
            128,-34964103,128,-42435336,128,-34964198,128,5259260,128,22036472,
            128,-34964157,128,22036472,128,-34964159,128,5259260,128,22036470,
            128,17842168,128,-34964163,128,22036470,128,17842167,128,-34964166,
            128,-42435367,128,-34964205,128,21495810,128,-28835837,128,-34964178,
            128,-28262406,128,-30359559,128,-12550136,128,-32899070,128,-44875785,
            128,-28835839,128,-64487413,128,-44401920,128,-34964193,128,21495809,
            128,-28835838,128,-42435340,128,-34964197,128,21495809,128,-42435344,
            128,13138686,54,54,54,54,58,58,128,22036477,128,-2097099,128,
            19529737,128,30425085,128,17694722,128,-34095107,128,13746172,
            128,22036479,128,-2097099,128,19628041,128,30425087,128,17793026,
            128,-34095105,128,13746174,128,-44843016,128,4719114,128,-42435422,
            128,13138686,54,54,54,128,-66584574,128,-42435352,128,5259259,
            128,-29559041,128,5259261,128,20805374,128,-44336384,128,4719115,
            128,-42435435,128,30965004,128,22575871,128,29360136,128,22576393,
            128,20641531,128,20870908,128,4784630,128,-42369907,128,22738682,
            128,-4095966,128,18380544,128,30964480,128,-46006263,128,0,128,
            30965011,128,22575871,128,29360136,128,22576400,128,-29690116,
            128,-31918340,128,-29427971,128,4784630,128,-42369922,128,-15141125,
            128,20641528,128,20870905,128,4784631,128,-42369927,128,22738679,
            128,-4095966,128,-31918335,128,25720064,128,30964480,128,-46006263,
            128,0,128,30965003,128,22575871,128,30965002,128,22576392,128,
            5259260,128,22036474,128,-28262407,128,-34964252,128,22036473,
            128,-28262407,128,-34964255,128,-44401406,128,0,128,0,128,30965003,
            128,22575871,128,30965002,128,22576392,128,5259260,128,22036472,
            128,-28262405,128,-34964266,128,22036472,128,-28262406,128,-34964269,
            128,-44401406,128,0,128,0,128,0,128,0,128,0,128,0,128,-31949570,
            128,1604858,128,18382074,128,30964480,128,21495809,128,31490055,
            128,17596417,128,22575871,128,-44303616,128,-64487422,128,-44401920,
            128,-28262403,128,-30359556,128,-14647288,128,-10945298,128,-28262401,
            128,-30359554,128,-14647288,128,-10945301,128,5259259,128,22036477,
            128,-28835839,128,5554168,128,-42369779,128,5554170,128,-42369786,
            128,-10978076,128,-27756799,128,-15173375,128,-34964253,128,-42435332,
            128,-10978081,128,-27756799,128,-15140607,128,-34964258,128,-42435332,
            128,5554170,128,-42369786,128,-10978088,128,-27724031,128,-15173375,
            128,-34964265,128,-42435332,128,-10978093,128,-27724031,128,-15140607,
            128,-34964270,128,-42435332⊱;

START:  PRINTTEXT(⊰BOUD MATS⊱);
        SPACE(100); PRINTTEXT(⊰bijlage 2. blz 1.⊱); NLCR;NLCR;

        comment nu eerst de arrays vullen;
        for j:= 1 step 1 until 50 do
        begin   v[j]:=j+1; for i:= 1 step 1 until 40 do
                begin   p[i,j]:=i+2×j; q[i,j]:=3×i+j×0.2 end
        end;

        PRINTTEXT(⊰ TEST1. op vecvec⊱); NLCR;
        PRINT(algolvecvec(1,50,0,v,v)); PRINT(vecvec(1,50,0,v,v)); NLCR; NLCR;

        PRINTTEXT(⊰ TEST2. op mattam⊱); NLCR;
        PRINT(algolmattam(20,50,3,2,p,q));
        PRINT(mattam(20,50,3,2,p,q)); NLCR; NLCR;
```

```
        PRINTTEXT(⊰ TEST3. op tijdsduur⊱); NLCR; t:=time;
        for k:=1,k+1 while k≤10000 do x:=algolmattam(1,50,3,2,p,q); z:=time;
        for k:=1,k+1 while k≤10000 do x:=mattam(1,50,3,2,p,q); y:=time;
        for k:=1,k+1 while k≤10000 do; x:=time;
        x:=x-y; y:=y-z-x; z:=z-t-x;
        PRINTTEXT(⊰duur in algol⊱); FLOT(4,3,z); NLCR;
        PRINTTEXT(⊰duur in code⊱); FLOT(4,3,y); NLCR;
        PRINTTEXT(⊰verhouding der snelheden⊱); FIXT(4,2,z/y); NLCR;NLCR;
        FIXT(4,2,time); NLCR;NLCR;
        PRINTTEXT(⊰ TEST4. Op foutmelding.
        De nu volgende foutmelding is opzettelijk in het programma opgenomen⊱);
        TELETEXT(⊰ VOLGENDE FOUTMELDING IS OPZET   BOUD⊱);
        x:=vecvec(1,4,0,v,p)
end
```

```
BOUD MATS

 TEST1. OP VECVEC
        +45525                  +45525

 TEST2. OP MATRAM
+.3041100000006₁₀+  5 +.3041100000006₁₀+  5

 TEST3. OP TIJDSDUUR
DUUR IN ALGOL+.3105₁₀+  3
DUUR IN CODE+.5929₁₀+  2
VERHOUDING DER SNELHEDEN   +5.24

 +37/.88

 TEST4. OP FOUTMELDING.
       DE NU VOLGENDE FOUTMELDING IS OPZETTELIJK IN HET PROGRAMMA OPGENOMEN
ER 523     419              -0
```

```
'BEGIN''MACRO'FILL:
        'BEGIN'L
L:
M[FILLPNT]:     :L[-1]
        0
FILLPNT[0]:     [FILLPNT+1]
L[0]:
        'END',

'MACRO'FILL2(TT,KK):            ''BEGIN UITBREIDING
        'BEGIN' L               ''TT=ALGOL-IDENTIFIER
L:                              ''KK=AANTAL SYMBOLEN
M[FILLPNT]:     :L[-1]
        0
FILLPNT[0]:     [FILLPNT+1]
M[TEXTPNT]:     :TT
                KK
TEXTPNT[0]:     [TEXTPNT+2]
L[0]:
        'END',                  ''EINDE UITBREIDING

'MACRO'IDI:
        'BEGIN'
        [   6 ]
        FILL
        'END',

'MACRO'TTP:
        'BEGIN'
        [   7 ]
        FILL
        'END',

'MACRO'TSR:
        'BEGIN'
        [  21 + [ 1 × '1000000' ]]
        FILL
        'SKIP'1
        'END',

'MACRO'TSI:
        'BEGIN'
        [  22 + [ 1 × '1000000' ]]
        FILL
        'SKIP'1
        'END',

'MACRO'TSB:
        'BEGIN'
        [  23 + [ 1 × '1000000' ]]
        FILL
        'SKIP'1
        'END',

'MACRO'TFSU:
        'BEGIN'
        [  25 ]
        FILL
        'END',
```

```
'MACRO'TSL:
        'BEGIN'
        [ 26 ]
        FILL
        'END',

'MACRO'TFSL:
        'BEGIN'
        [ 27 ]
        FILL
        'END',

'MACRO'TCST:
        'BEGIN'
        [ 28 ]
        FILL
        'END',

'MACRO'STSR:
        'BEGIN'
        [ 29 ]
        FILL
        'END',

'MACRO'STSI:
        'BEGIN'
        [ 30 ]
        FILL
        'END',

'MACRO'SSTSI:
        'BEGIN'
        [ 31 ]
        FILL
        'END',

'MACRO'STSB:
        'BEGIN'
        [ 32 ]
        FILL
        'END',

'MACRO'STFSU:
        'BEGIN'
        [ 34 ]
        FILL
        'END',

'MACRO'CRV:
        'BEGIN'
        [ 53 ]
        FILL
        'END',

'MACRO'CIV:
        'BEGIN'
        [ 54 ]
        FILL
        'END',
```

```
'MACRO'CBV:
        'BEGIN'
        [  55 ]
        FILL
        'END',

'MACRO'CLV:
        'BEGIN'
        [  57 ]
        FILL
        'END',

'MACRO'CEN:
        'BEGIN'
        [  58 ]
        FILL
        'END',

'MACRO'CLPN:
        'BEGIN'
        [  59 ]
        FILL
        'END',

'MACRO'ENTIER:
        'BEGIN'
        [  78 ]
        FILL
        'END',

'MACRO'SQRT:
        'BEGIN'
        [  79 ]
        FILL
        'END',

'MACRO'EXP:
        'BEGIN'
        [  80 ]
        FILL
        'END',

'MACRO'LN:
        'BEGIN'
        [  81 ]
        FILL
        'END',

'MACRO'PEND:
        'BEGIN'
        [  82 ]
        FILL
        'END',

'MACRO'TRV(TT,KK):              ''BEGIN UITBREIDING
        'BEGIN'
        [ 83+['1000']]          ''+D9=512 ALS
        FILL2(TT,KK)            '' HERKENNINGS BIT
```

```
        'END',                  ''ALGOL IDENTIFIER

'MACRO'TIV(TT,KK):
        'BEGIN'
        [ 84+['1000']]
        FILL2(TT,KK)
        'END',

'MACRO'TBV(TT,KK):
        'BEGIN'
        [ 88+['1000']]
        FILL2(TT,KK)
        'END',

'MACRO'TAK(TT,KK):
        'BEGIN'
        [ 92+['1000']]
        FILL2(TT,KK)
        'END',

'MACRO'STR(TT,KK):
        'BEGIN'
        [ 94+['1000']]
        FILL2(TT,KK)
        'END',

'MACRO'STI(TT,KK)
        'BEGIN'
        [ 95+['2001000']]
        FILL2(TT,KK)
        'SKIP'2
        'END',

'MACRO'SSTI(TT,KK):
        'BEGIN'
        [ 96+['1000']]
        FILL2(TT,KK)
        'END',

'MACRO'STB(TT,KK):
        'BEGIN'
        [ 97+['1001000']]
        FILL2(TT,KK)
        'SKIP'1
        'END',

'MACRO'JU1(TT,KK):
        'BEGIN'
        [ 103+['1001000']]
        FILL2(TT,KK)
        'SKIP'1
        'END',

'MACRO'IJU(TT,KK):
        'BEGIN'
        [ 104+['1000']]
        FILL2(TT,KK)
        'END',
```

```
'MACRO'IJU1(TT,KK):
        'BEGIN'
        [ 105+['1000']]
        FILL2(TT,KK)
        'END',

'MACRO'SUBJ(TT,KK):
        'BEGIN'
        [ 108+['1000']]
        FILL2(TT,KK)
        'END',

'MACRO'ISUBJ(TT,KK):
        'BEGIN'
        [ 109+['1000']]
        FILL2(TT,KK)
        'END',                  ''EINDE UITBREIDING

'MACRO'SIN:
        'BEGIN'
        [ 129 ]
        FILL
        'END',

'MACRO'COS:
        'BEGIN'
        [ 130 ]
        FILL
        'END',

'MACRO'ARCTAN:
        'BEGIN'
        [ 131 ]
        FILL
        'END',

'MACRO'PREAD:
        'BEGIN'
        [ 132 ]
        FILL
        'END',

'MACRO'PREHEP:
        'BEGIN'
        [ 140 ]
        FILL
        'END'

START:  B=:STACK
        A=:TEXTAR               ''UITBREIDING
        TEXTPNT=A               ''EINDE UITBREIDING
        SUBC(:PREPR)
        JUMP(3)
        :LIPR1
        :LIPCO
        :NEWLI
        F=:HEAD
        S=:PICTHEAD
        A=LIPR1
```

```
        SUBC(:EXDIT)
        SUBC(:CLOCK)
        SUBC(:PREPR)    ''OPEN TAPEPUNCH
        GOTO(:L1)
        :TAPU1
        :FLEXO
        :BPEND
LIST:   'SKIP'2000
FILLPNT:        :LIST
EMPTYPNT:       :LIST
PUNCHPNT:       :FIRST
FIRSTSW:        1
POSPNT:         0
STACK:  'SKIP'200
TEXTAR:         'SKIP'64        ''UITBREIDING
TEXTPNT:        :TEXTAR         ''IDEM
D18M1:  '7777777'

L1:     SUBC(:TAPEFEED)
        A=116   ''NLCR
        SUBC(:DOUBLE)
        A=125   '' |
        SUBC(TAPU1)
        A=53
        SUBC(TAPU1)
        A=125
        SUBC(LIPR1)
        A=114
        SUBC(LIPR1)
        A=53
        SUBC(LIPR1)
L2:     A=PUNCHPNT      ''ADR. OF NEXT INSTR.
        S=EMPTYPNT
        S=MS    ''ADR. OF NEXT STANDARD MACRO
        U,A-S,Z         ''EQUAL?
        A=MA
        Y,GOTO(:NOCODE)
        MC=A
        S=128   ''CODE
        SUBC(:PUNCHNBR)
        S=MC[-1]        ''INSTRUCTION
        SUBC(:PUNCHNBR)
INCRPNT: A=1            ''INCREASE PUNCHPOINTER
        PLUSA(PUNCHPNT)
        A-:LAST,P       ''ALL DONE?
        N,GOTO(:L2)
        A=125   ''PUNCH /
        SUBC(TAPU1)
        A=54
        SUBC(TAPU1)
        A=125
        SUBC(LIPR1)
        A=114
        SUBC(LIPR1)
        A=54
        SUBC(LIPR1)
        A=39    ''PUNCH SEMICOLON

        SUBC(:DOUBLE)
```

```
        A=116   ''PUNCH NLCR
        SUBC(:DOUBLE)
        SUBC(:TAPEFEED)
        A=TAPU1
        SUBC(:CONCL)
        F=:TAIL
        S=:PICTTAIL
        A=LIPR1
        SUBC(:EXDIT)
        SUBC(:CLOCK)
        A=LIPR1
        SUBC(:CONCL)
        SUBC(:STOP)

NOCODE:         S=A
        RUA(18)
        PUNCHPNT+A
        S'x'D18M1
        U,S'x'512,Z     ''UITBREIDING, GEEN BIT?
        N,S'x'511       ''VERWYDER BIT
        SUBC(:PUNCHNBR)
        N,SUBC(:INSERT)         ''REACTIE OP BIT,EINDE UITBREIDING
        A=1
        EMPTYPNT+A
        GOTO(:INCRPNT)

        'BEGIN'L1
TAPEFEED:       A=300
        COUNT=A
L1:     A=127
        SUBC(TAPU1)
        REPP(:L1)
        GOTOR(MC[-1])
        'END'
PUNCHNBR:       'BEGIN'L1,L2
        A=FIRSTSW,P     ''FIRST NBR
        Y,FIRSTSW=-B
        N,A=40  ''NO,PUNCH,
        N,SUBC(:PUNCHSL)
        A=POSPNT        ''MORE THAN 60 SLS?
        A-60,P
        Y,A=116         ''YES,PUNCH NLCR
        Y,SUBC(:PUNCHSL)
        Y,A=0
        Y,POSPNT=A
        S=S,P   ''NEGATIVE?
        N,S=-S  ''TAKE ABSOLUTE VALUE
        N,A=47  ''PUNCH-
        N,SUBC(:PUNCHSL)
        MC=-B   ''ENDMARKER
L1:     A=0     ''STACK ALL SYMBOLS
        DIVAS(10)
        MC=A
        S=S,Z   ''READY?
        N,GOTO(:L1)
L2:     A=MC[-1],P      ''ENDMARKER?
        Y,SUBC(:PUNCHSL)        ''NO,PUNCH SYMBOL
        Y,GOTO(:L2)
        GOTOR(MC[-1])
```

```
        'END'

PUNCHSL:        MC=S
        SUBC(:DOUBLE)
        A=1
        POSPNT+A
        S=MC[-1]
        GOTOR(MC[-1])

DOUBLE:         MC=A
        SUBC(TAPU1)     ''PUNCH SYMBOL
        A=MC[-1]
        U,A-116,Z       ''NLCR?
        SUBC(LIPR1)     ''PRINT SYMBOL
        N,GOTOR(MC[-1])
        A=12
        COUNT=A
TAB:    A=60            ''TAB NA NLCR
        SUBC(:DOUBLE)
        REPP(:TAB)
        GOTOR(MC[-1])

INSERT:         A=TEXTPNT       ''UITBREIDING-KK
                S=MA[1]
                S+:PICTZ
                MC=S
                G=MA            '':TT
                MC=F
                A=40            ''KOMMA
                SUBC(:PUNCHSL)
                S=M[B-3]
                F=M[B-2]
                A=LIPR1
                SUBC(:EXDIT)
                F=MC[-2]
                S=M[B-1]
                A=TAPU1
                SUBC(:EXDIT)
                S=MC[-1]
                S-:PICTZ
                POSPNT+S
                A=2
                TEXTPNT+A
                GOTOR(MC[-1])

PICTZ:  'PICT'A(0)
        'PICT'A(1)
        'PICT'A(2)
        'PICT'A(3)
        'PICT'A(4)
        'PICT'A(5)
        'PICT'A(6)
        'PICT'A(7)
        'PICT'A(8)
        'PICT'A(9)
        'PICT'A(10)     ''EINDE UITBREIDING
```

Bijlage 3  blz.9

```
CLOCK:  'BEGIN'PP       ''UITBREIDING
        G=M[220]
        S=:PP
        A=LIPR1
        SUBC(:EXDIT)
        GOTOR(MC[-1])
PP:     'PICT'BZ(6)P9(2)
        'END' CLOCK     ''EINDE UITBREIDING
HEAD:   '('BOUD PCP')'  ''HEADING ALLEEN OP RD
PICTHEAD:       'PICT' A(8)
TAIL:   '('EINDE')'     ''ENDMARKER ALLEEN OP RD
PICTTAIL:       'PICT' A(5)
```