stichting
mathematisch
centrum

MC

AFDELING NUMERIEKE WISKUNDE NN 8/76 APRIL

J.KOK & K. DEKKER (SAMENSTELLERS)

# VERGELIJKING VAN REKENTIJDEN, VERSLAG VAN DE VERGELIJKING VAN DOOR VERSCHILLENDE VERTALERS VERTAALDE PROGRAMMA'S OP CONTROL DATA CYBER 73 COMPUTERS

2e boerhaavestraat 49 amsterdam

BIBLIOTHEEK MATHEMATISCH CENTRUM
——AMSTERDAM——

*Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.*

*The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.*

---

AMS(MOS) subject classification scheme (1970): 68-00, 68A20, 65-01

---

Vergelijking van rekentijden,
Verslag van de vergelijking van door verschillende vertalers vertaalde programma's op Control Data CYBER 73 computers

door

J.Kok & K. Dekker (samenstellers).

KORTE SAMENVATTING

In dit verslag wordt de efficiëntie vergeleken van enige vertalers en van verschillende vertaalde programma's, geschreven in hogere programmeertalen.

De programma's betreffen voor het merendeel het oplossen van problemen uit de numerieke wiskunde.

TREFWOORDEN: *Vertalers, efficiëntie, hogere programmeertalen, numerieke wiskunde.*

VOORWOORD

In 1975 werden door deelnemers aan de Werkgroep Numerieke Programmatuur resultaten bijeengebracht voor een vergelijking van rekentijden, nodig voor het oplossen van bepaalde problemen met behulp van programma's, geschreven in verschillende programmeertalen. De bijdragen zijn geleverd door de volgende personen:

| | |
|---|---|
| P.A. Beentjes | Mathematisch Centrum |
| K. Dekker | Universiteit van Amsterdam en Mathematisch Centrum |
| T.J. Dekker | Universiteit van Amsterdam |
| P. van Emde Boas | Universiteit van Amsterdam en Mathematisch Centrum |
| J.M. van Kats | Academisch Computer Centrum, Utrecht |
| J. Kok | Mathematisch Centrum |
| H.W. Lenstra Jr. | Universiteit van Amsterdam |
| T.H.P. Reymer | Universiteit van Amsterdam |
| P.H.M. Wolkenfelt | Universiteit van Amsterdam. |

In dit rapport wordt een overzicht gegeven van de resultaten, gevolgd door een beschrijving van de behandelde problemen. In een Appendix worden de teksten van de programma's gegeven.

J. Kok.

INHOUD

# 1. INLEIDING

Voor de vergelijking van de efficiëntie van vertalers werden rekentijden van complete programma's verzameld. De programma's in de verschillende talen bevatten een veelheid van opdrachten, zoals rekenkundige bewerkingen, procedure-aanroepen, gebruik van geheugenplaatsen, in- en uitvoer, enz., die van probleem tot probleem zowel qua aantal als qua aard sterk kunnen verschillen. Per probleem zijn de opdrachten van de programma's in de verschillende talen wel vergelijkbaar, te meer daar er naar gestreefd is voor hetzelfde probleem dezelfde algoritme te gebruiken.

Voor een aantal problemen zijn (ook) numerieke procedures uit programmatheken gebruikt. Waar dit de bibliotheek NUMAL (zie ref. [9]) betreft, zijn in de overige talen als regel equivalenten van de gebruikte NUMAL-procedures gemaakt. Daarnaast is enkele malen ook een FORTRAN-subroutine-bibliotheek gebruikt, zodat in die gevallen een bepaald probleem ook mogelijk met verschillende algoritmen is opgelost.

De gekozen problemen laten zich naar hun afkomst in enige groepen delen, waartussen zich ook de grote verschillen voordoen in de reeds genoemde verhouding tussen de aantallen opdrachten in een programma. Deze groepen zijn:

| | |
|---|---|
| gewone kwadratuur en differentiaalvergelijkingen | (1,2) |
| numerieke lineaire algebra | (3,4,5) |
| non-numerieke problemen (veelal recursief opgelost) | (6,7,8) |
| aritmetiek | (9) |
| in- en uitvoer | (10) . |

De gebruikte talen zijn ALGOL 60, ALGOL 68, FORTRAN, PASCAL en SIMULA. Voor elk probleem worden in de Appendix alle programma-teksten gegeven, met dien verstande, dat teksten van procedures uit een bibliotheek niet gegeven worden, maar wel de teksten van in andere talen gemaakte equivalente procedures of subroutines. Deze programmatheken zijn ACCULIB [1], IMSL [6], MSL [8] en NUMAL [9].

Alle programma's zijn uitgevoerd op de Control Data CYBER 73-28 van het Academisch Rekencentrum Amsterdam (SARA) of op de Control Data CYBER 73-26 van het Academisch Computer Centrum Utrecht (ACCU).

## 2. GEBRUIKTE VERTALERS EN OPTIES

De gebruikte vertalers zijn de op de gebruikte Control Data installaties als volgt geïdentificeerde versies (zie [2], [3], [4], [5], [7] en [10]):

ALGOL 60

a) versie 3.1, level 376A,
b) versie 4, level 13.

ALGOL 68

versie I, level 1.0.8.

FORTRAN

a) FTN,
b) MNF.

PASCAL

PASCAL 6000 - 3.4.

SIMULA

SIMULA 67.

Van deze vertalers is voor de vergelijking van executietijden de belangrijkste optie, dat bij het indiceren in arrays al of niet gecontroleerd wordt of een door de subscript(s) aangewezen element in het voor het array gereserveerde stuk geheugenruimte ligt. De twee mogelijkheden zijn:

a) controle op de grenzen van elke subscript (kortweg subscript-grenzen-controle genoemd),
b) controle op de grenzen van het array als geheel (array-grenzen-controle); in dit geval kunnen subscripts fout zijn, terwijl toch de door de foute subscripts aangewezen geheugenplaats binnen de grenzen van het array ligt, zodat er geen foutmelding wordt gegeven.

Andere gebruikte opties zijn:

a) optimalisatie van for-loops (alleen relevant bij ALGOL 60 waar de for-loop erg duur is),

TABEL 2.1

| Taal | Versie | Optie, Aanduiding | | Verklaring |
|---|---|---|---|---|
| ALGOL 60 | 3.1 | array-controle | Q | geen array-grenzen-controle, wel optimalisatie van for-loops |
| | | | afwezig | array-grenzen-controle |
| ALGOL 60 | 4 | array-controle | C = 3 | subscript-grenzen-controle (checkon in comment) |
| | | | C = 0 | geen array-controle |
| | | parameter correspondentie | X(=1) | integer-real-correspondentie toegestaan |
| | | | X = 0 | correspondentie verboden |
| | | optimalisatie van code | O = 2 | optimalisatie bij procedure-aanroep, subscripting en for-loops |
| | | | O = 0 | geen optimalisatie |
| ALGOL 68 | | array-controle | A | geen array-controle |
| | | | afwezig | subscript-grenzen-controle |
| FORTRAN | FTN | array-controle | D | array-grenzen-controle (bij ARRAYS-directive) |
| | | | afwezig | geen array-controle |
| FORTRAN | MNF | array-controle | D | array-grenzen-controle (bij TRACE SUBSCRIPTS directive) |
| | | | afwezig | geen array-controle |
| PASCAL | | array-controle | T+ | subscript-grenzen-controle |
| | | | T- | geen array-controle |
| | | PMD-tabel | P+ | bijhouden van tabel van namen voor Post Mortem Dump |
| | | | P- | geen PMD-tabel |
| SIMULA | SIMULA 67 | array-controle | N | geen array-controle |
| | | | afwezig | array-grenzen-controle |

b) optimalisatie van code bij procedure-aanroepen,
c) toestaan van integer-real-correspondentie bij actuele-formele parameters van procedures (relevant bij programma's met veel procedure-aanroepen),
d) bijhouden van tabel van namen van variabelen in programma voor de Post Mortem Dump (alleen PASCAL).

Daar de verschillende vertalers verschillende mogelijkheden (waaronder opties) hebben, zijn de resultaten van de programma's lang niet altijd te vergelijken. Dit hangt mede af van de wensen van de gebruikers. Hier wordt ook niet ingegaan op de moeiten, die de gebruikers soms moeten nemen (zoals veranderingen in programma's) om bepaalde vertaler-opties werkelijk te verkrijgen. Voor een overzicht van alle opties en de wijze van verkrijgen van die opties zij verwezen naar de verschillende Manuals.

Tabel 2.1 geeft een verklaring van de gebruikte opties.

## 3. OVERZICHT RESULTATEN

Daar probleem 1 (kwadratuur) op twee manieren is opgelost (met recursieve procedures en met niet-recursieve procedures en subroutines) bevat tabel 3.1 11 kolommen voor de verschillende problemen. In elke kolom zijn opgenomen:

compilatietijd (CT),
executietijd (ET).

De in tabel 3.1 vermelde tijden zijn uitgedrukt in seconden, en nauwkeurig in tienden van seconden.

Als regel zijn de compilatie-tijden afgelezen uit day-file-boodschappen van de vertalers (of door SUMMARY-aanroepen), terwijl de rekentijden tijdens executie zijn gemeten door aanroepen van een klok-procedure ("clock").

In veel gevallen zijn programma's niet met alle mogelijke opties uitgevoerd, omdat geen essentieel nieuwe resultaten werden verwacht. Dit is het geval bij de array-controle-optie voor programma's die weinig met arrays werken, maar ook bij het gebruik van bibliotheken: de combinatie [FTN, IMSL] zal geen grote verschillen geven met [MNF, IMSL] zoals ook [PASCAL, IMSL] geen

TABEL 3.1

| | 1 recursief | | 1 niet-recursief | | 2 | | 3 | | 4 | | 5 | | 6 | | 7 | | 8 | | 9 | | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | CT | ET | CT | ET | CT | ET | CT | ET | CT | ET | CT | ET | CT | ET | CT | ET | CT | ET | CT | ET | CT | ET |
| ALGOL 60,V3,- | 3.5 | 55.4 | 3.6 | 47.4 | 5.3 | 92.1 | 3.8 | 36.6 | 3.0 | 13.5 | 8.6 | 69.7 | 1.9 | 8.1 | 2.6 | 173.2 | 4.2 | 91.7 | 0.8 | 29.6 | 2.2 | 154.5 |
| V3,Q | 2.8 | 45.3 | 3.4 | 41.9 | 4.1 | 63.7 | 3.3 | 22.3 | 2.5 | 6.6 | 7.6 | 45.6 | 1.7 | 6.1 | 2.4 | 132.0 | 3.8 | 61.8 | | | 2.5 | 149.6 |
| V3,-,NUMAL | | | | | | | 1.9 | 13.2 | | | 2.1 | 25.4 | | | | | | | | | | |
| V4,C=3 | 4.5 | 61.8 | 4.8 | 47.5 | 6.1 | 119.5 | 5.4 | 55.7 | 3.6 | 23.8 | 7.6 | 114.8 | 2.8 | 12.1 | 3.5 | 230.4 | 5.3 | 119.7 | | | | |
| V4,O=2 | | | | | | | | | 3.7 | 6.5 | | | | | | | | | | | | |
| V4,X=0 | 4.4 | 56.9 | | | 5.6 | 64.7 | | | | | | | | | | | 5.0 | 55.1 | | | | |
| V4,- | 4.5 | 61.4 | 4.6 | 42.8 | 5.8 | 74.5 | 5.1 | 32.2 | 3.6 | 11.1 | 7.8 | 60.9 | 2.7 | 7.9 | 3.6 | 110.1 | 5.0 | 55.1 | | | 2.6 | 102.7 |
| ALGOL 68 | | | | | | | | | | | | | | | | | | | | | | |
| - | 15.0 | 26.0 | | | 20.9 | 128.1 | 17.8 | 45.8 | 12.9 | 16.0 | 23.9 | 79.5 | 7.8 | 8.7 | 10.7 | 192.8 | 15.7 | 84.0 | 3.6 | 30.4 | 12.0 | 185.0 |
| A | 14.7 | 25.7 | | | 19.5 | 80.4 | 17.4 | 24.8 | 12.6 | 8.0 | 23.7 | 48.0 | 7.6 | 6.1 | 10.6 | 113.5 | 15.4 | 44.8 | | | | 175.7 |
| FORTRAN | | | | | | | | | | | | | | | | | | | | | | |
| FTN,- | | | 4.8 | 15.9 | 3.0 | 25.4 | | | 1.5 | 2.1 | 3.1 | 7.7 | | | 1.6 | 31.7 | | | 0.7 | 17.6 | 1.7 | 9.1 |
| FTN,D | | | | | | | | | | | 3.9 | 52.4 | | | | | | | | | | |
| FTN,-,MSL | | | | | | | | | | | 1.4 | 11.6 | | | | | | | | | | |
| MFN,- | | | | | | | | | | | 3.1 | 8.4 | | | | | | | | | 2.4 | 13.2 |
| MFN,D | | | | | | | | | | | 3.3 | 9.3 | | | | | | | | | | |
| MFN,-,IMSL | | | | | | | | | | | 2.1 | 34.4 | | | | | | | | | | |
| PASCAL | | | | | | | | | | | | | | | | | | | | | | |
| - | 4.2 | 16.9 | 4.4 | 17.6 | 4.6 | 39.0 | 4.1 | 11.4 | 3.2 | 6.3 | 9.4 | 23.6 | 2.7 | 1.8 | 3.4 | 48.1 | 4.2 | 27.2 | 2.5 | 20.1 | 3.7 | 15.1 |
| T-,P- | 4.1 | 17.0 | 4.3 | 17.0 | 4.4 | 33.3 | 4.0 | 8.5 | 3.0 | 4.5 | 9.3 | 15.2 | 2.8 | 1.3 | 3.3 | 35.1 | 4.1 | 20.3 | | | 3.5 | 13.3 |
| -,IMSL | | | | | | | 3.2 | 5.4 | | | 3.0 | 36.5 | | | | | | | | | | |
| SIMULA | | | | | | | | | | | | | | | | | | | | | | |
| - | | | | | | | | | 3.6 | 16.8 | | | | | | | | | | | | |
| N | | | | | | | | | 3.6 | 15.6 | | | | | | | | | | | | |

grote afwijking geeft.

In enkele andere gevallen is afgezien van het maken van een FORTRAN-programma, omdat in de andere talen recursieve procedures gebruikt werden.

Bij alle programma's, behalve die voor probleem 10, is de rekentijd, nodig voor in- en/of uitvoer gering ten opzichte van de totale rekentijd en daarom niet apart opgenomen (in enkele gevallen werd uitvoer pas gegeven nadat de totale rekentijd met een klok-procedure was gemeten).

Voor een uitgebreidere vergelijking van tijden voor invoer wordt verwezen naar het rapport van VAN DE WIJGAART & STORM-VAN ESSEN [12], waarin ook de verschillende mogelijkheden van formats worden vergeleken.

## 4. CONCLUSIES

Vergelijken van de rekentijden is slechts mogelijk per probleem, zoals ook in hoofdstuk 1 is vermeld. Daardoor is het uitvoeren op verschillende machines (die overigens weinig verschillen) geen bezwaar, omdat per probleem altijd op dezelfde computer gewerkt is. Van de opgegeven tijden moet nog opgemerkt worden, dat deze een nauwkeurigheid van hoogstens 5% hebben.

Bij het vergelijken van de vertaaltijden moet op de volgende punten gelet worden:

a) Voor verschillende talen wordt van de mogelijkheden van de talen gebruik gemaakt, waardoor de teksten soms veel van elkaar afwijken.
b) Voor problemen, die met bibliotheek-procedures worden opgelost, zijn alleen de aanroepende hoofdprogramma's vertaald.

Over de vertaaltijden kan dan geconcludeerd worden:

a) Per taal hangt de vertaaltijd van programma's vrijwel lineair af van de lengte van de programma's.
b) Het gebruik van de in tabel 2.1 genoemde opties van een vertaler heeft weinig invloed op de vertaaltijd gehad.
c) De FTN-vertaler is bijna altijd het snelst, behalve bij het kwadratuur-probleem, waar voor het berekenen van een meervoudige integraal twee

gelijke subroutine-teksten met verschillende namen vertaald moesten worden.

d) De ALGOL 68-vertaler heeft verreweg de meeste tijd nodig. (Van de in februari 1976 geïnstalleerde nieuwe versie van de vertaler wordt door SARA vermeld, dat de vertaaltijden met 40% verminderd zijn. Deze verandering is door ons voor enkele gevallen geverifiëerd).

De executietijden van de diverse programma's kunnen niet zonder meer met elkaar vergeleken worden. Men moet hierbij rekening houden met de volgende punten:

a) Programma's, waarin veel met arrays wordt gewerkt, zijn voor verschillende talen het best vergelijkbaar als zonder array-grenzen-controle of subscript-grenzen-controle wordt gewerkt.

b) Mét array-controle zijn onderling vergelijkbaar:
   (i) ALGOL 60 versie 3, FORTRAN (FTN en MNF), SIMULA (allen array-grenzen-controle),
   (ii) ALGOL 60 versie 4, ALGOL 68, PASCAL (allen subscript-grenzen-controle).

c) Naast het onder b) genoemde onderscheid moet bij arrays nog op de volgende verschillen van de talen gelet worden:
   (i) in FORTRAN is de ondergrens van elke array-index altijd 1,
   (ii) PASCAL gebruikt statische arrays, wat belangrijk is bij het doorgeven van arrays aan procedures,
   (iii) alleen in ALGOL 68 kunnen de controles ook uitgevoerd worden voor delen van bestaande arrays.

d) De verschillende vertalers hebben vaak ook nog andere, voor de gebruikers soms erg nuttige opties, die hier buiten beschouwing zijn gebleven, omdat ze bij andere vertalers vaak geen equivalenten hebben.

e) Bij het gebruik van procedure-(subroutine-)bibliotheken kan het volgende opgemerkt worden:
   (i) de bibliotheekprocedures leverden ongeveer dezelfde resultaten (binnen de gevraagde nauwkeurigheid),
   (ii) bij de gebruikte FORTRAN-subroutines wordt *niet* op array-grenzen gecheckt,
   (iii) bij NUMAL-procedures (ALGOL 60, versie 3) wordt *wel* op array-grenzen gecontroleerd, maar de code van vector- en matrix-operaties

is geoptimaliseerd,

(iv) probleem 5 wordt door de IMSL-bibliotheek-subroutine met een totaal andere algoritme opgelost (singuliere wàardenontbinding) dan door de andere programma's en bibliotheekprocedures (Householder-orthogonalisatie met kolom-verwisselen).

Rekening houdend met het bovengenoemde komen wij tot de volgende slotsommen:

a) Bij veel array-gebruik, zonder grenzen-controle: FTN en MNF geven de snelste programma's, daarna PASCAL, vervolgens op grote afstand ALGOL 60 versie 3 en ALGOL 68, en daarna ALGOL 60 versie 4.
b) Bij veel array-gebruik, met array-grenzen-controle: MNF is zeer snel, op grote afstand FTN en ALGOL 60 versie 3.
c) Bij veel array-gebruik, met subscript-grenzen-controle: PASCAL is snel (ook in vergelijking met FTN), daarna ALGOL 68, daarna ALGOL 60 versie 4.
d) Bij veel procedure-aanroepen (probleem 1 recursief, 6, 8): PASCAL is snel, gevolgd door ALGOL 68 en soms ALGOL 60 versie 3, daarna ALGOL 60 versie 4. FORTRAN kan niet aan deze vergelijking meedoen.
e) Overige programma's met veel rekenwerk: FORTRAN en PASCAL zijn snel, ALGOL 60 en ALGOL 68 zijn veel langzamer.
f) Veel invoer en uitvoer (probleem 10): FTN is het snelst, daarna MNF en PASCAL, daarna ALGOL 60 versie 4 (!), vervolgens ALGOL 60 versie 3, tenslotte ALGOL 68.
g) Conclusies over de rekentijden van SIMULA-programma's zijn niet verantwoord, o.m. omdat niet van de speciale mogelijkheden van deze taal gebruik is gemaakt.

## 5. VERSLAGEN

In dit hoofdstuk worden korte omschrijvingen van de behandelde problemen en bijzonderheden over de uitvoering gegeven. Voor de geleverde vertaaltijden en executietijden wordt verwezen naar tabel 3.1.

## 5.1. K. DEKKER: *Kwadratuur*

De integralen

$$\int_0^1 \frac{\sin(m\pi x)}{m\pi x}\,dx$$

en

$$\int_0^1\int_0^1 \cos(m\pi yx)\,dxdy, \qquad m = 1,3,10,30,$$

worden berekend met behulp van de integratieformules, zoals deze worden toegepast in de procedure *qadrat* (zie [9], sectie 4.2.1).

In de programma's zijn zowel recursieve als niet-recursieve varianten van deze procedure gebruikt.

De berekening van deze integralen vereiste ongeveer $10^5$ cosinus/sinus-evaluaties.

## 5.2. P.A. BEENTJES, K. DEKKER: *Drie-lichamen-probleem*

De baan van een satelliet in de omgeving van twee hemellichamen wordt beschreven door een stelsel tweede-orde differentiaalvergelijkingen. Door middel van een Runge-Kutta-methode, zoals deze is toegepast in de procedure *rke* (zie [9], sectie 5.2.1.1.1.1), werd dit stelsel vergelijkingen opgelost.

## 5.3. T.J. DEKKER, T.H.P. REYMER: *Oplossen van stelsels lineaire vergelijkingen*

Voor matrices A van orde $n$ ($n$ = 10,40,70,100) wordt het stelsel lineaire vergelijkingen Ax = b opgelost door middel van Cholesky-decompositie, zoals deze wordt toegepast in de procedure *chldecsol 1* (zie [9], sectie 3.1.1.1.1.2.3). De elementen van A worden gegeven door

$$A_{ij} = \frac{1}{i+j-1}, \qquad \text{voor } j \neq i,$$

$$A_{ii} = \frac{1}{2i-1} + \frac{1}{10}, \qquad i = 1(1)n, \quad j = 1(1)n.$$

## 5.4. J.M. VAN KATS: *Tridiagonalisatie van een symmetrische matrix*

Door middel van Householder-transformaties wordt een symmetrische matrix op tridiagonaalgedaante gebracht. De orde van de matrix is 25 en het proces wordt 10 maal uitgevoerd. Het belangrijkste deel van de verwerking is array-access.

De ALGOL 60-procedure is gepubliceerd in J.H. WILKINSON & C. REINSCH [11], waarnaar ook de FORTRAN-versie uit EISPACK is gemaakt.

## 5.5. J. KOK: *Lineaire kleinste-kwadraten stelsels*

Opgelost worden de vergelijkingen Ax = b, waarin A een n * m matrix, voor n = 10(10)60, m = 1(1)15. A is een segment van de Hilbert-matrix, d.w.z.

$$a_{ij} = \frac{1}{i+j-1} \quad , \qquad i = 1(1)n, \quad j = 1(1)m.$$

Voor het rechterlid geldt:

$$b_i = \sum_{j=1}^{m} \frac{j}{i+j-1} \, , \qquad i = 1(1)n.$$

De programma's maken gebruik van verschillende bibliotheken of van uitgeschreven analogi van de procedure *lsqortdecsol* (zie [9], sectie 3.1.1.2.1.2). De belangrijkste werkzaamheid is array-subscripting.

## 5.6. T.J. DEKKER, T.H.P. REYMER: *Sorteren van een rij getallen*

Een rij reële getallen wordt met een recursief sort/merge proces naar grootte geordend. In dit proces wordt de rij eerst in twee deelrijen van (bijna) gelijke lengte gesplitst. Daarna worden beide deelrijen afzonderlijk gesorteerd, en tenslotte samengevoegd tot één gesorteerde rij. De bewerkelijkheid van dit proces is $n.^2\log n$ (n = lengte van de rij). In de programma's is voor de lengte 2000 gekozen.

5.7. K. DEKKER, H.W. LENSTRA JR.: *Getaltheoretische functie*

Bij elk priemgetal p kan een functie

$$L_p : \mathbb{N} \longrightarrow \{\underline{\text{true}}, \underline{\text{false}}\}$$

gedefinieerd worden door:

$$L_p(k) = \underline{\text{true}} \Longleftrightarrow \exists i,\ k = i^2 \ \underline{\text{mod}}\ p,$$
$$\underline{\text{false}} \Longleftrightarrow \forall i,\ k \neq i^2 \ \underline{\text{mod}}\ p.$$

We definiëren nu f(p) als de lengte van de langste rij opeenvolgende natuurlijke getallen, waarvoor $L_p$ dezelfde waarde aanneemt. In het programma wordt de waarde van deze functie voor alle priemgetallen kleiner dan 5000 berekend.

5.8. P.H.M. WOLKENFELT: *Met vierkanten gevulde rechthoek*

Het programma bepaalt alle oplossingen om een gegeven rechthoek (n.l. met afmetingen 32 * 33) te verdelen in vierkanten van gehele, maar onderling verschillende afmetingen (*perfect squared rectangle*). De verwerking wordt gekenmerkt door de vele recursieve procedure-aanroepen (waarin alle mogelijkheden om een vierkant in de rechthoek te plaatsen worden afgezocht) en door vrij veel array-subscripting. De in- en uitvoer is te verwaarlozen.

5.9. P. VAN EMDE BOAS: *Integer deling*

Voor $1 \leq i \leq j \leq 1000$ worden j en $(i * j)/i$ vergeleken. Indien de uitkomst ongelijk is, wordt een melding gemaakt van een onjuist uitgevoerde deling. In de praktijk blijkt geen foute deling op te treden.

## 5.10. J. KOK: *Afdrukken van pentomino-oplossingen*

368 oplossingen van een pentomino-probleem (om een 4 * 15-rechthoek met de 12 pentomino's te vullen) worden ingelezen, en vervolgens geprint, waarbij de scheidingslijnen tussen de pentomino's worden gerepresenteerd door streepjes op de regeldrukker. De executietijd bestaat in hoofdzaak uit in- en uitvoertijd.

### VERANTWOORDING

De problemen zijn gekozen en voor het merendeel ook geprogrammeerd en getest door de auteurs van de in hoofdstuk 5 gegeven verslagen.

Daarnaast hebben ook meerdere programmeurs van het Mathematisch Centrum meegewerkt aan het gereedmaken van programma's en van de in de Appendix geleverde teksten.

De bijdragen werden door K. Dekker en mij verzameld en samengevat.

J. Kok.

### REFERENTIES

[1] ACCULIB. ACCU-programma-bibliotheek (ALGOL en FORTRAN).
Academisch Computer Centrum Utrecht (1975).

[2] ALGOL 60 Reference Manual, version 3.
Control Data Cyber 70 Computer Systems.

[3] ALGOL 60 Reference Manual Cyber 70 Series Version 4.
Control Data Cyber 70 Computer Systems.

[4] ALGOL 68 Version I Reference Manual.
Control Data Cyber 70 Series Computer Systems (1975).

[5] FORTRAN Extended Reference Manual (met: MNF-Minnesota FORTRAN insertions).
Control Data Cyber 70 Computer Systems.

[6] IMSL Library 3 Reference Manual.
International Mathematical and Statistical Libraries, Inc. (1975).

[7] JENSEN, K. & N. WIRTH: PASCAL User Manual and Report.
Springer Verlag (1974).

[8] MSL. Math. Science Library.
Control Data Corporation (1971).

[9] NUMAL. A Library of Numerical Procedures in ALGOL 60.
Mathematisch Centrum, Amsterdam (1974).

[10] SIMULA Reference Manual.
Control Data 6000 Computer Systems (1973).

[11] WILKINSON, J.H. & C. REINSCH: Linear Algebra.
Springer Verlag (1971).

[12] WIJGAART, C. VAN DE, & L. STORM-VAN ESSEN: Leestijden, een onderzoek naar de efficiency van een zestal op SARA beschikbare compilers bij het inlezen van data.
Technisch Centrum Faculteit der Sociale Wetenschappen, Universiteit van Amsterdam (1975).

APPENDIX: Programmateksten

Hieronder volgen voor elk probleem de programma's in (voor zover aanwezig) ALGOL 60, ALGOL 68, FORTRAN, PASCAL.

Er zijn geen bijna-doublures opgenomen. Dit betreft de volgende gevallen:

a) Per probleem worden geen aparte teksten gegeven voor programma's in ALGOL 60 versie 3 of versie 4 en SIMULA. De geringe verschillen betreffen telkens externe referenties en comments (versie 4) en namen van output-statements (SIMULA).
b) Voor enkele problemen is het uitgangspunt van de oplossingsmethode een NUMAL-procedure. In die gevallen zijn de procedure-teksten ook apart vertaald met ALGOL 60 versie 3, en met ALGOL 60 versie 4, maar er wordt alleen een ALGOL 60-hoofdprogramma afgedrukt.
c) De teksten van programma's voor FTN en MNF verschillen alleen in de directive-kaarten voor de compiler-optie.
d) Enkele problemen zijn mede opgelost door aanroepen van een bibliotheek-subroutine. De hoofdprogramma's (in FORTRAN of PASCAL) verschillen dan alleen voor wat betreft de subroutine-aanroep (naam van de subroutine en lijst van parameters) en de externe referentie (in PASCAL).

De volgorde van de programmateksten is die van hoofdstuk 5.

```
"BEGIN" "COMMENT" INTEGRATIE MET QADRAT;
    "REAL" "PROCEDURE" QADRAT(X, A, B, FX, E);
    "VALUE" A, B; "REAL" X, A, B, FX; "ARRAY" E;
    "BEGIN" "REAL" F0, F2, F3, F5, F6, F7, F9,
         F14, V, W, HMIN, HMAX, RE, AE;

       "REAL" "PROCEDURE" LINT(X0, XN, F0, F2, F3, F5, F6, F7, F9, F14);
       "REAL" X0, XN, F0, F2, F3, F5, F6, F7, F9, F14;
       "BEGIN" "REAL" H, XM, F1, F4, F8, F10, F11, F12, F13;
          XM:= (X0 + XN) / 2; H:= (XN - X0) / 32; X:= XM + 4 * H;
          F8:= FX; X:= XN - 4 * H; F11:= FX; X:= XN - 2 * H; F12:= FX;
          V:= 0,330580178199226 * F7 + 0,173485115707338 * (F6 + F8) +
          0,321105426559972*(F5 + F9) + 0,135007708341042 * (F3 + F11)
          + 0,165714514228223*(F2 + F12) + 0,593971460638127"- 1 * (F0
          + F14); X:= X0 + H; F1:= FX; X:= XN - H; F13:= FX;
          W:= 0,260652441323638 * F7 + 0,239063283351431 * (F6 + F8) +
          0,263062635477467*(F5 + F9) + 0,218681931383057 * (F3 + F11)
          + 0,275789764664284"- 1 * (F2 + F12) + 0,105575010053846* (F1
          + F13) + 0,157119426059518"- 1 * (F0 + F14);
          "IF" ABS(H) < HMIN "THEN" E[3]:= E[3] + 1;
          "IF" ABS(V - W) < ABS(W) * RE + AE "OR" ABS(H) < HMIN
          "THEN" LINT:= H * W "ELSE"
          "BEGIN" X:= X0 + 6 * H; F4:= FX; X:= XN - 6 * H; F10:= FX;
             V:= 0,245673430150304* F7 + 0,255786258286921* (F6 + F8) +
             0,228526063690406*(F5 + F9) + 0,500557131555861"- 1*(F4 +
             F10) + 0,177946487756780*(F3 + F11)+0,584014599032140"- 1
             * (F2 + F12) + 0,874830942871332"- 1 * (F1 + F13) +
             0,189642078648079"- 1 * (F0 + F14);
             LINT:= "IF" ABS(V - W) < ABS(V) * RE + AE "THEN" H * V
                                                    "ELSE"
             LINT(X0, XN, F0, F1, F2, F3, F4, F5, F6, F7) - LINT(XN,
             XM, F14, F13, F12, F11, F10, F9, F8, F7)
          "END"
       "END" LINT;

       HMAX:= (B - A) / 16; "IF" HMAX=0 "THEN"
       "BEGIN" QADRAT:= 0; "GOTO" RETURN "END";
       RE:= E[1]; AE:= 2 * E[2] / ABS(B - A); E[3]:= 0;
       HMIN:= ABS(B - A) * RE; X:= A; F0:= FX;
       X:= A + HMAX; F2:= FX; X:= A + 2 * HMAX; F3:= FX;
       X:= A + 4 * HMAX; F5:= FX; X:= A + 6 * HMAX; F6:= FX;
       X:= A + 8 * HMAX; F7:= FX; X:= B - 4 * HMAX; F9:= FX; X:= B;
       F14:= FX;
       QADRAT:= LINT(A, B, F0, F2, F3, F5, F6, F7, F9, F14) * 16;
       RETURN;
    "END" QADRAT;

    "REAL" X,Y,XL,PI,TIME;
    "INTEGER" SING,M,TF,TG,EVALF,EVALG;
    "ARRAY" E1,E2[1:3];
    "REAL" "PROCEDURE" F(Y); "VALUE" Y; "REAL" Y;
    "BEGIN" TF:=TF+1; F:=COS(XL*Y) "END";
    "REAL" "PROCEDURE" G(X); "VALUE" X; "REAL" X;
    "BEGIN" XL:="X*PI; TG:=TG+1;
        "IF" SING=2 "THEN" G:=QADRAT(Y,0,1,F(Y),E2) "ELSE"
        "IF" XL=0 "THEN" G:=1 "ELSE" G:=SIN(XL)/XL
    "END";

    TIME:=CLOCK; OUTPUT(61,"("("TIJD=")",3Z,3D,/")",TIME);
    PI:=4*ARCTAN(1); EVALF:=EVALG:=0;
    "FOR" SING:=1,2 "DO"
    "FOR" M:=1,3,10,30 "DO"
    "FOR" E1[1]:="-3,"-6,"-9 "DO"
    "BEGIN" E1[2]:=E1[1]; E2[1]:=E2[2]:=E1[1]/10;
        TF:=TG:=0;
        OUTPUT(61,"("("M=")",ZD,"(" TOL=")",D,D"-D,"(" INTEGRAAL=")",
        -Z.9D,"(" F EVAL=")",5ZD,"(" G EVAL=")",2ZD,/")",M,E1[1],
        QADRAT(X,0,1,G(X),E1),TF,TG);
        EVALF:=EVALF+TF; EVALG:=EVALG+TG;
        "IF" E1[1]="-9 "THEN" OUTPUT(61,"(/")");
        "IF" E1[1]="-9 "AND" M=30 "THEN" OUTPUT(61,"(/")");
    "END";
    OUTPUT(61,"("("AANTAL EVALUATIES F:")",6ZD,/,
    "("AANTAL EVALUATIES G:")",6ZD,/,"("VERBRUIKTE TIJD    :")",6ZD,3D,
    /")",EVALF,EVALG,CLOCK-TIME)
"END"
```

```
'BEGIN'
    'PROC' F=('REAL' Y) 'REAL': (TF+:=1; COS(XL*Y));
    'PROC' G=('REAL' X) 'REAL': (TG+:=1; XL:=M*X*PI);
        'IF' SING=2 'THEN' QADRAT(0.0,1.0,F,F2) 'ELIF'
             XL=0 'THEN' 1.0 'ELSE' SIN(XL)/XL 'FI');
    'INT' EVALF:=0,EVALG:=0,TF,TG,M,SING;
    'REAL' XL,TIME; 'REAL' PI=4*ARCTAN(1);
    'PROC' QADRAT=('REAL' A,B, 'PROC' ('REAL') 'REAL' F,
    'REF' [ ] 'REAL' E) 'REAL';
    'BEGIN' 'REAL' HMAX:=(B-A)/16;
      'IF' HMAX=0 'THEN' 0 'ELSE'
        'REAL' V7=0.330580178199226,     V6=0.173485115707338,
               V5=0.321105426559972,     V3=0.135007708341042,
               V2=0.165714514228223,     V0=0.393971460638127E-1,
               W7=0.260652441323638,     W6=0.239063283351431,
               W5=0.263062635477407,     W3=0.218681931383057,
               W2=0.275789764664284E-1, W1=0.105575010053846,
               W0=0.157119426059518E-1,
               U7=0.245673430150304,     U6=0.255786258286921,
               U5=0.228526063690406,     U4=0.500557131555861E-1,
               U3=0.177946487736780,     U2=0.584014599032140E-1,
               U1=0.874830942871332E-1, U0=0.189642078648079E-1;
        'REAL' RE=E[1],AE=2*E[2]/'ABS'(B-A),HMIN='ABS'(B-A)*RE;
        'REAL' H,V,W,X0:=A,XN:=B;
        'REAL' L0:=F(A),L2:=F(A+HMAX),L3:=F(A+2*HMAX),L5:=F(A+4*HMAX),
           L6:=F(A+6*HMAX),L7:=F(A+8*HMAX),R5:=F(B-4*HMAX),R0:=F(B);
        E[3]:=0;
        'PROC' LINT=('REF''REAL'X0,XN,L0,L2,L3,L5,L6,L7,R5,R0) 'REAL';
        'BEGIN' H:=(XN-X0)/32;
            'REAL' XM:=(X0+XN)/2, R6:=F(XM+4*H),R3:=F(XN-4*H),
                   R2:=F(XN-2*H);
            V:=V7*L7+V6*(L6+R6)+V5*(L5+R5)+V3*(L3+R3)+V2*(L2+R2)+
               V0*(L0+R0);
            'REAL' L1:=F(X0+H),R1:=F(XN-H);
            W:=W7*L7+W6*(L6+R6)+W5*(L5+R5)+W3*(L3+R3)+W2*(L2+R2)+
               W1*(L1+R1)+W0*(L0+R0);
            'IF' 'ABS'(H)<HMIN 'THEN' E[3]+:=1; H*W 'ELIF'
               'ABS'(V-W)<'ABS'(W)*RE+AE 'THEN' H*W 'ELSE'
               'REAL' L4:=F(X0+6*H),R4:=F(XN-6*H);
               V:=U7*L7+U6*(L6+R6)+U5*(L5+R5)+U4*(L4+R4)+U3*(L3+R3)+
                  U2*(L2+R2)+U1*(L1+R1)+U0*(L0+R0);
               'IF' 'ABS'(V-W)<'ABS'(V)*RE+AE 'THEN' H*V 'ELSE'
                  LINT(X0,XM,L0,L1,L2,L3,L4,L5,L6,L7)+
                  LINT(XN,XM,R0,R1,R2,R3,R4,R5,R6,L7)
               'FI'
            'FI'
        'END';
        LINT(X0,XN,L0,L2,L3,L5,L6,L7,R5,R0)*16
      'FI'
    'END';
    TIME:=CLOCK; PRINT(("TIJD=",FIXED(TIME,10,3),NEWLINE));
    [1:4] 'INT' MW:=(1,3,10,30);
    [1:3] 'REAL' ER:=(1E-3,1E-6,1E-9),E1,E2;
    'FOR' K 'TO' 2
    'DO' 'FOR' J 'TO' 4
        'DO' 'FOR' I 'TO' 3
            'DO' E1[1]:=E1[2]:=ER[I]; E2[1]:=E2[2]:=ER[I]/10;
                TF:=0;TG:=0;M:=MW[J];SING:=K;
                PRINT(("M=",WHOLE(M,3)," TOL=",FLOAT(E1[1],8,1,3),
                " INTEGRAAL=",FIXED(QADRAT(0,1,G,E1),11,9)," F EVAL=",
                WHOLE(TF,6)," G EVAL=",WHOLE(TG,4),NEWLINE));
                EVALF+:=TF; EVALG+:=TG;
                'IF' I=3 'THEN' PRINT(NEWLINE);
                    'IF' J=4 'THEN' PRINT(NEWLINE) 'FI'
                'FI'
            'OD'
        'OD'
    'OD';
    PRINT((" AANTAL EVALUATIES F:",WHOLE(EVALF,7),NEWLINE));
    PRINT((" AANTAL EVALUATIES G:",WHOLE(EVALG,7),NEWLINE));
    PRINT((" VERBRUIKTE TIJD    :",FIXED(CLOCK-TIME,9,3),NEWLINE))
'END'
```

```
FUNCTION QADRAT;
    VAR L0,L2,L3,L5,L6,L7,R5,R0,V,W,HMIN,HMAX,AE,RE: REAL;
    FUNCTION LINT(X0,XN,L0,L2,L3,L5,L6,L7,R5,R0: REAL): REAL;
        CONST V7=0,3305801781992226;    V6=0,1734851157073381;
              V5=0,3211054265599721;    V3=0,1350077083410421;
              V2=0,1657145142282231;    V0=0,3939714606381271E-1;
              W7=0,2606524413236381;    W6=0,2390632833514311;
              W5=0,2630626354774671;    W3=0,2186819313830571;
              W2=0,2757897646642841E-1; W1=0,1055750100538461;
              W0=0,1571194260595181E-1;
              U7=0,2456734301503041;    U6=0,2557862582869211;
              U5=0,2285260656904061;    U4=0,5005571315558611E-1;
              U3=0,1779464877367801;    U2=0,5840145990321401E-1;
              U1=0,8748309428713321E-1;
              U0=0,1896420786480791E-1;
        VAR   H,XM,L1,L4,R6,R4,R3,R2,R1: REAL;
        BEGIN XM:=(X0+XN)/2; H:=(XN-X0)/32; R6:=F(XM+4*H);
           R3:=F(XN-4*H); R2:=F(XM-2*H);
           V:=V7*L7+V6*(L6+R6)+V5*(L5+R5)+V3*(L3+R3)+V2*(L2+R2)+
               V0*(L0+R0);
           L1:=F(X0+H); R1:=F(XM-H);
           W:=W7*L7+W6*(L6+R6)+W5*(L5+R5)+W3*(L3+R3)+W2*(L2+R2)+
               W1*(L1+R1)+W0*(L0+R0);
           IF ABS(H)<HMIN THEN E[3]:=E[3]+1;
           IF (ABS(V-W) < ABS(W)*RE+AE) OR (ABS(H)<HMIN) THEN LINT:=H*W
           ELSE
           BEGIN L4:=F(X0+6*H); R4:=F(XN-6*H);
              V:=U7*L7+U6*(L6+R6)+U5*(L5+R5)+U4*(L4+R4)+U3*
                 (L3+R3)+U2*(L2+R2)+U1*(L1+R1)+U0*(L0+R0);
              IF ABS(V-W) < ABS(V)*RE+AE THEN LINT:=H*V ELSE
              LINT:=LINT(X0,XM,L0,L1,L2,L3,L4,L5,L6,L7)+
                    LINT(XM,XN,R0,R1,R2,R3,R4,R5,R6,L7)
           END
        END;

    BEGIN HMAX:=(B-A)/16; IF HMAX>0 THEN
       BEGIN RE:=E[1]; AE:=2*E[2]/ABS(B-A); E[3]:=0;
             HMIN:=ABS(B-A)*RE; L0:=F(A); L2:=F(A+HMAX);
             L3:=F(A+2*HMAX); L5:=F(A+4*HMAX); L6:=F(A+6*HMAX);
             L7:=F(A+8*HMAX); R5:=F(B-4*HMAX); R0:=F(B);
             QADRAT:=LINT(A,B,L0,L2,L3,L5,L6,L7,R5,R0)*16
       END
    END;
```

```
"BEGIN" "COMMENT" INTEGRATIE MET QADRAT, STAPEL I,P,V, RECURSIEF;
    "REAL" XL,PI,TIME;
    "INTEGER" SING,M,TF,TG,EVALF,EVALG;
    "ARRAY" E1,E2[1:3];
    "REAL" "PROCEDURE" F(Y); "VALUE" Y; "REAL" Y;
    "BEGIN" TF:=TF+1; F:=COS(XL*Y) "END";
    "REAL" "PROCEDURE" G(X); "VALUE" X; "REAL" X;
    "BEGIN" XL:=M*X*PI; TG:=TG+1;
        "IF" SING=2 "THEN" G:=QADRAT(0,1,F,E2) "ELSE"
        "IF" XL=0 "THEN" G:=1 "ELSE" G:=SIN(XL)/XL
    "END";

    "REAL" "PROCEDURE" QADRAT(A,B,F,E);
    "VALUE" A,B; "REAL" A,B; "REAL" "PROCEDURE" F; "ARRAY" E;
    "BEGIN" "REAL" H,X0,XM,XN,L0,L1,L2,L3,L4,L5,L6,L7,R0,R1,R2,R3,R4,R5,
                   R6,V,W,HMIN,HMAX,ABSH,AE,RE,INT;
        "COMMENT" "CHECKON" PAR0,PAR1,PAR2,PAR3,PAR4,PAR5,PAR6,PAR7,
                            PAR8,PAR9;
        "ARRAY" PAR0,PAR1,PAR2,PAR3,PAR4,PAR5,PAR6,PAR7,PAR8,PAR9[1:45];
        "INTEGER" P;
        "BOOLEAN" NEW;
        HMAX:=(B-A)/16; INT:=0; "IF" HMAX=0 "THEN" "GOTO" EIND;
        RE:=E[1]; AE:=2*E[2]/ABS(B-A); E[3]:=0;
        HMIN:=ABS(B-A)*RE; L0:=F(A); L2:=F(A+HMAX);
        L3:=F(A+2*HMAX); L5:=F(A+4*HMAX); L6:=F(A+6*HMAX);
        L7:=F(A+8*HMAX); R5:=F(B-4*HMAX); R0:=F(B); X0:=A; XN:=B;
        P:=1; NEW:="FALSE";
    NEXT; "IF" NEW "THEN"
        "BEGIN" X0:=PAR0[P]; XN:=PAR1[P]; L0:=PAR2[P]; L2:=PAR3[P];
            L3:=PAR4[P]; L5:=PAR5[P]; L6:=PAR6[P]; L7:=PAR7[P];
            R5:=PAR8[P]; R0:=PAR9[P];
        "END"; P:=P-1; NEW:="TRUE";
        XM:=(X0+XN)/2; H:=(XN-X0)/32; ABSH:=ABS(H); R6:=F(XM+4*H);
        R3:=F(XM-4*H); R2:=F(XN-2*H);
        V:=0,3305801781992 26*L7+0,1734851157073 38*(L6+R6)+
           0,3211054265599 72*(L5+R5)+0,1350077083410 42*(L3+R3)+
           0,1637145142282 23*(L2+R2)+0,3939714606381 27"-1*(L0+R0);
        L1:=F(X0+H); R1:=F(XN-H);
        W:=0,2606524413236 38*L7+0,2390632833514 31*(L6+R6)+
           0,2630626354774 67*(L5+R5)+0,2186819313830 57*(L3+R3)+
           0,2757897646642 84"-1*(L2+R2)+0,1055750100538 46*(L1+R1)+
           0,1571194260595 18"-1*(L0+R0);
        "IF" ABSH<HMIN "THEN" E[3]:=E[3]+1;
        "IF" ABS(V-W)<ABS(W)*RE+AE "OR" ABSH<HMIN "THEN" INT:=ABSH*W+INT
        "ELSE"
        "BEGIN" L4:=F(X0+6*H); R4:=F(XN-6*H);
            V:=0,2456734301503 04*L7+0,2557862582869 21*(L6+R6)+
               0,2285260636904 06*(L5+R5)+0,5005571315558 61"-1*(L4+R4)+
               0,1779464877367 80*(L3+R3)+0,5840145990321 40"-1*(L2+R2)+
               0,8748309428713 32"-1*(L1+R1)+,1896420786480 79"-1*(L0+R0);
            "IF" ABS(V-W)<ABS(V)*RE+AE "THEN" INT:=ABSH*V+INT "ELSE"
            "BEGIN" P:=P+1; PAR0[P]:=XN; PAR1[P]:=XM; PAR2[P]:=R0;
                PAR3[P]:=R1; PAR4[P]:=R2; PAR5[P]:=R3; PAR6[P]:=R4;
                PAR7[P]:=R5; PAR8[P]:=R6; PAR9[P]:=L7;
                P:=P+1; NEW:="FALSE"; R0:=L7; R5:=L6; L7:=L5; L6:=L4;
                L5:=L3; L3:=L2; L2:=L1; XN:=XM
            "END";
        "END";
        "IF" P>0 "THEN" "GOTO" NEXT;
    EIND: QADRAT:=INT*16
    "END" QADRAT;

    TIME:=CLOCK; OUTPUT(61,"("("TIJD=")",3Z,3D,/")",TIME);
    PI:=4*ARCTAN(1); EVALF:=EVALG:=0;
    "FOR" SING:=1,2 "DO"
    "FOR" M:=1,3,10,30 "DO"
    "FOR" E1[1]:="-3,"-6,"-9 "DO"
    "BEGIN" E1[2]:=E1[1]; E2[1]:=E2[2]:=E1[1]/10;
        TF:=TG:=0;
        OUTPUT(61,"("("M=")",ZD,"(" TOL=")",D,D"-D,"(" INTEGRAAL=")",
        -Z,9D,"(" F EVAL=")",5ZD,"(" G EVAL=")",2ZD,/")",M,E1[1],
        QADRAT(0,1,G,E1),TF,TG);
        EVALF:=EVALF+TF; EVALG:=EVALG+TG;
        "IF" E1[1]="-9 "THEN" OUTPUT(61,"(/")");
        "IF" E1[1]#"-9 "AND" M=30 "THEN" OUTPUT(61,"(/")");
    "END";
    OUTPUT(61,"("("AANTAL EVALUATIES F:")",6ZD,/,
    "("AANTAL EVALUATIES G:")",6ZD,/,"("VERBRUIKTE TIJD    :")",6ZD,3D,
    /")",EVALF,EVALG,CLOCK-TIME)
"END"
```

```
      PROGRAM QUADRAT(OUTPUT)
      EXTERNAL G
      COMMON M,SING,PI,E2(3) $ INTEGER M,SING $ REAL PI
      COMMON /TELLER/TL,TX   $ INTEGER TL,TX
      REAL E1(3),ER(3),TIME
      INTEGER MW(4),EVALF,EVALG
    5 FORMAT(* TIJD=*,F10.3)
      TIME=SECOND(CP) $ PRINT 5,TIME
      PI=4.*ATAN(1.)
      MW(1)=1 $ MW(2)=3 $ MW(3)=10 $ MW(4)=30
      ER(1)=1E-3 $ ER(2)=1E-6 $ ER(3)=1E-9
      EVALF=0 $ EVALG=0
      DO 40 SING=1,2
      DO 30 J=1,4
      DO 20 I=1,3
      TL=0 $ TX=0 $ M=MW(J)
      E1(1)=ER(I) $ E1(2)=ER(I) $ E2(1)=ER(I)/10 $ E2(2)=ER(I)/10
      PRINT 10,M,E1(1),QADRAT1(0.,1.,E1,G),TL,TX
   10 FORMAT(* M=*,I2,* TOL=*,E6.1,* INTEGRAAL=*,F12.9,* F EVAL=*,I6,
     $       * G EVAL=*,I3)
   20 IF (I.EQ.3) PRINT 50
   30 IF (J.EQ.4) PRINT 50
   40 CONTINUE
   50 FORMAT(1H )
      PRINT 60,EVALF,EVALG,SECOND(CP)-TIME
   60 FORMAT(* AANTAL EVALUATIES F:*,I7,/,* AANTAL EVALUATIES G:*,I7,/,
     $       * VERBRUIKTE TIJD    :*,F10.3)
      END

      REAL FUNCTION F(Y) $ REAL Y
      COMMON /TELLER/TL,TX $ INTEGER TL,TX
      COMMON /FG/XL $ REAL XL
      TL=TL+1 $ F=COS(XL*Y)
      END

      REAL FUNCTION G(X) $ REAL X
      COMMON M,SING,PI,E2(3) $ INTEGER M,SING $ REAL PI
      COMMON /TELLER/TL,TX   $ INTEGER TL,TX
      COMMON /FG/XL $ REAL XL
      EXTERNAL F
      TX=TX+1 $ XL=M*X*PI
      IF (SING.EQ.1) 10,20
   10 IF (XL.EQ.0) 11,12
   11 G=1. $ GOTO 30
   12 G=SIN(XL)/XL $ GOTO 30
   20 G=QADRAT2(0.,1.,E2,F)
   30 CONTINUE
      END

      REAL FUNCTION QADRAT1(A,B,E,F) $ EXTERNAL F $ REAL A,B,E(3)
      REAL V0,V2,V3,V5,V6,V7,W0,W1,W2,W3,W5,W6,W7,
     $     U0,U1,U2,U3,U4,U5,U6,U7,RE,AE,HMIN,ABSH,HMAX,INT
      REAL L0,L1,L2,L3,L4,L5,L6,L7,R0,R1,R2,R3,R4,R5,R6
      REAL PAR0(45),PAR1(45),PAR2(45),PAR3(45),PAR4(45),PAR5(45),
     $     PAR6(45),PAR7(45),PAR8(45),PAR9(45)
      INTEGER P
      LOGICAL NEW
      DATA (V7=0.33058017819226), (V6=0.1734851157073 38),
     $     (V5=0.32110542655997 2), (V3=0.135007708341042),
     $     (V2=0.16571451022822 3), (V0=0.393971460638127E-1),
     $     (W7=0.26065244132363 8), (W6=0.23906328335143 1),
     $     (W5=0.26306263547746 7), (W3=0.21868193138305 7),
     $     (W2=0.27578976466428 4E-1), (W1=0.105575010053846),
     $     (W0=0.15711942605951 8E-1),
     $     (U7=0.24567343015030 4), (U6=0.25578625828692 1),
     $     (U5=0.22852606369040 6), (U4=0.50055713155386 1E-1),
     $     (U3=0.17794687736780), (U2=0.58401459903214 0E-1),
     $     (U1=0.87483094287133 2E-1), (U0=0.18964207864807 9E-1)
      HMAX=(B-A)/16. $ INT=0.
      IF (HMAX.LE.0.) GOTO 100
      RE=E(1) $ AE=2*E(2)/ABS(B-A) $ E(3)=0. $ HMIN=ABS(B-A)*RE
      L0=F(A) $ L2=F(A+HMAX) $ L3=F(A+2.*HMAX) $ L5=F(A+4.*HMAX)
      L6=F(A+6.*HMAX) $ L7=F(A+8.*HMAX) $ R5=F(B-4.*HMAX) $ R0=F(B)
      X0=A $ XN=B $ P=1 $ NEW=.T.
   30 IF (NEW) GOTO 40
      X0=PAR0(P) $ XN=PAR1(P) $ L0=PAR2(P) $ L2=PAR3(P) $ L3=PAR4(P)
      L5=PAR5(P) $ L6=PAR6(P) $ L7=PAR7(P) $ R5=PAR8(P) $ R0=PAR9(P)
   40 P=P-1 $ NEW=.F.
      XM=(X0+XN)/2. $ H=(XN-X0)/32. $ ABSH=ABS(H) $ R6=F(XM+4.*H)
      R3=F(XN-4.*H) $ R2=F(XN-2.*H) $ L1=F(X0+H) $ R1=F(XM-H)
```

```
      V=V7*L7+V6*(L6+R6)+V5*(L5+R5)+V3*(L3+R3)+V2*(L2+R2)+V0*(L0+R0)
      W=W7*L7+W6*(L6+R6)+W5*(L5+R5)+W3*(L3+R3)+W2*(L2+R2)+W1*(L1+R1)
     $  +W0*(L0+R0)
      IF (ABSH .LT. HMIN) E(3)=E(3)+1.
      IF (ABS(V-W) .LT. ABS(W)*RE+AE .OR. ABSH .LT. HMIN) 50,60
50    INT=ABSH*W+INT $ GOTO 70
60    L4=F(X0+6.*H) $ R4=F(XN-6.*H)
      V=U7*L7+U6*(L6+R6)+U5*(L5+R5)+U4*(L4+R4)+U3*(L3+R3)+U2*(L2+R2)
     $  +U1*(L1+R1)+U0*(L0+R0)
      IF (ABS(V-W) .LT. ABS(V)*RE+AE) 61,62
61    INT=ABSH*V+INT $ GOTO 70
62    P=P+1 $ PAR0(P)=XN $ PAR1(P)=XM $ PAR2(P)=R0 $ PAR3(P)=R1
      PAR4(P)=R2 $ PAR5(P)=R3 $ PAR6(P)=R4 $ PAR7(P)=R5 $ PAR8(P)=R6
      PAR9(P)=L7
      P=P+1 $ NEW=.T. $ R0=L7 $ R5=L6 $ L7=L5 $ L6=L4 $ L5=L3 $ L3=L2
      L2=L1 $ XN=XM
70    IF (P .GT. 0) GOTO 30
100   QADRAT1=INT*16.
      END
      REAL FUNCTION QADRAT2(A,B,E,F) $ EXTERNAL F $ REAL A,B,E(3)
      REAL V0,V2,V3,V5,V6,V7,W0,W1,W2,W3,W5,W6,W7,
     $     U0,U1,U2,U3,U4,U5,U6,U7,RE,AE,HMIN,ABSH,HMAX,INT
      REAL L0,L1,L2,L3,L4,L5,L6,L7,R0,R1,R2,R3,R4,R5,R6
      REAL PAR0(45),PAR1(45),PAR2(45),PAR3(45),PAR4(45),PAR5(45),
     $     PAR6(45),PAR7(45),PAR8(45),PAR9(45)
      INTEGER P
      LOGICAL NEW
      DATA (V7=0.330580178199226), (V6=0.173485115707338),
     $     (V5=0.321105426559772), (V3=0.135007708341042),
     $     (V2=0.165714514228223), (V0=0.393971460638127E-1),
     $     (W7=0.260652441323638), (W6=0.239063283351431),
     $     (W5=0.263062635477467), (W3=0.218681931383057),
     $     (W2=0.275789764664284E-1), (W1=0.105575010053846),
     $     (W0=0.157119426059518E-1),
     $     (U7=0.245673430150304), (U6=0.255786258286921),
     $     (U5=0.228526063690406), (U4=0.500557131555861E-1),
     $     (U3=0.177946487736780), (U2=0.584014599032140E-1),
     $     (U1=0.874830942871332E-1), (U0=0.189642078648079E-1)
      HMAX=(B-A)/16. $ INT=0.
      IF (HMAX.LE.0.) GOTO 100
      RE=E(1) $ AE=2*E(2)/ABS(B-A) $ E(3)=0. $ HMIN=ABS(B-A)*RE
      L0=F(A) $ L2=F(A+HMAX) $ L3=F(A+2.*HMAX) $ L5=F(A+4.*HMAX)
      L6=F(A+6.*HMAX) $ L7=F(A+8.*HMAX) $ R5=F(B-4.*HMAX) $ R0=F(B)
      X0=A $ XN=B $ P=1 $ NEW=.T.
30    IF (NEW) GOTO 40
      X0=PAR0(P) $ XN=PAR1(P) $ L0=PAR2(P) $ L2=PAR3(P) $ L3=PAR4(P)
      L5=PAR5(P) $ L6=PAR6(P) $ L7=PAR7(P) $ R5=PAR8(P) $ R0=PAR9(P)
40    P=P-1 $ NEW=.F.
      XM=(X0+XN)/2. $ H=(XN-X0)/32. $ ABSH=ABS(H) $ R6=F(XM+4.*H)
      R3=F(XM-4.*H) $ R2=F(XN-2.*H) $ L1=F(X0+H) $ R1=F(XN-H)
      V=V7*L7+V6*(L6+R6)+V5*(L5+R5)+V3*(L3+R3)+V2*(L2+R2)+V0*(L0+R0)
      W=W7*L7+W6*(L6+R6)+W5*(L5+R5)+W3*(L3+R3)+W2*(L2+R2)+W1*(L1+R1)
     $  +W0*(L0+R0)
      IF (ABSH .LT. HMIN) E(3)=E(3)+1.
      IF (ABS(V-W) .LT. ABS(W)*RE+AE .OR. ABSH .LT. HMIN) 50,60
50    INT=ABSH*W+INT $ GOTO 70
60    L4=F(X0+6.*H) $ R4=F(XN-6.*H)
      V=U7*L7+U6*(L6+R6)+U5*(L5+R5)+U4*(L4+R4)+U3*(L3+R3)+U2*(L2+R2)
     $  +U1*(L1+R1)+U0*(L0+R0)
      IF (ABS(V-W) .LT. ABS(V)*RE+AE) 61,62
61    INT=ABSH*V+INT $ GOTO 70
62    P=P+1 $ PAR0(P)=XN $ PAR1(P)=XM $ PAR2(P)=R0 $ PAR3(P)=R1
      PAR4(P)=R2 $ PAR5(P)=R3 $ PAR6(P)=R4 $ PAR7(P)=R5 $ PAR8(P)=R6
      PAR9(P)=L7
      P=P+1 $ NEW=.T. $ R0=L7 $ R5=L6 $ L7=L5 $ L6=L4 $ L5=L3 $ L3=L2
      L2=L1 $ XN=XM
70    IF (P .GT. 0) GOTO 30
100   QADRAT2=INT*16.
      END
```

```
PROGRAM QADTEST(OUTPUT);
    TYPE AR3=ARRAY[1:3] OF REAL;
    VAR  XL,PI,TIME: REAL;
         SING,M,TF,TG,EVALF,EVALG,I,J: INTEGER;
         E1,E2,ER: AR3;
         MW: ARRAY[1:4] OF INTEGER;
    FUNCTION QADRAT(A,B:REAL; VAR E:AR3; FUNCTION F:REAL):REAL; FORWARD;
    FUNCTION F(Y:REAL):REAL;
    BEGIN TF:=TF+1; F:=COS(XL*Y) END;
    FUNCTION G(X:REAL):REAL;
    BEGIN XL:=M*X*PI; TG:=TG+1;
       IF SING=2 THEN G:=QADRAT(0,1,E2,F) ELSE
       IF XL=0 THEN G:=1 ELSE G:=SIN(XL)/XL
    END;

    FUNCTION QADRAT;
        CONST V7=0,330580178199226;    V6=0,173485115707338;
              V5=0,321105426559972;    V4=0,135007708341042;
              V2=0,165714514228223;    V0=0,393971460638127E-1;
              W7=0,260652441323638;    W6=0,239063283351431;
              W5=0,263062354777467;    W3=0,218681931383057;
              W2=0,275789764664284E-1; W1=0,105575010053846;
              W0=0,157119426059518E-1;
              U7=0,245673430150304;    U6=0,255786258286921;
              U5=0,228526063690406;    U4=0,500557131555861E-1;
              U3=0,177946487736780;    U2=0,584014599032140E-1;
              U1=0,874830942871332E-1;
              U0=0,189642078648079E-1;
        VAR   H,X0,XN,XM,L0,L1,L2,L3,L4,L5,L6,L7,R0,R1,R2,R3,R4,R5,R6,
              V,W,HMIN,HMAX,ABSH,AE,RE,INT: REAL;
              PAR0,PAR1,PAR2,PAR3,PAR4,PAR5,PAR6,PAR7,PAR8,PAR9:
              ARRAY[1:45] OF REAL;
              P: INTEGER;
              NEW: BOOLEAN;
    BEGIN HMAX:=(B-A)/16; INT:=0; IF HMAX<=0 THEN P:=0 ELSE
       BEGIN RE:=E[1]; AE:=2*E[2]/ABS(B-A); E[3]:=0;
          HMIN:=ABS(B-A)*RE; L0:=F(A); L2:=F(A+HMAX);
          L3:=F(A+2*HMAX); L5:=F(A+4*HMAX); L6:=F(A+6*HMAX);
          L7:=F(A+8*HMAX); R5:=F(B-4*HMAX); R0:=F(B); X0:=A; XN:=B;
          P:=1; NEW:=FALSE;
       END;
       WHILE P>0 DO
       BEGIN IF NEW THEN
          BEGIN X0:=PAR0[P]; XN:=PAR1[P]; L0:=PAR2[P]; L2:=PAR3[P];
             L3:=PAR4[P]; L5:=PAR5[P]; L6:=PAR6[P]; L7:=PAR7[P];
             R5:=PAR8[P]; R0:=PAR9[P]
          END; P:=P-1; NEW:=TRUE;
          XM:=(X0+XN)/2; H:=(XN-X0)/32; ABSH:=ABS(H); R6:=F(XN+4*H);
          R3:=F(XN-4*H); R2:=F(XN-2*H);
          V:=V7*L7+V6*(L6+R6)+V5*(L5+R5)+V3*(L3+R3)+V2*(L2+R2)+
             V0*(L0+R0);
          L1:=F(X0+H); R1:=F(XN-H);
          W:=W7*L7+W6*(L6+R6)+W5*(L5+R5)+W3*(L3+R3)+W2*(L2+R2)+
             W1*(L1+R1)+W0*(L0+R0);
          IF ABSH<HMIN THEN E[3]:=E[3]+1;
          IF (ABS(V-W)<ABS(W)*RE+AE) OR (ABSH<HMIN) THEN INT:=ABSH*W+INT
          ELSE
          BEGIN L4:=F(X0+6*H); R4:=F(XN-6*H);
             V:=U7*L7+U6*(L6+R6)+U5*(L5+R5)+U4*(L4+R4)+U3*
                (L3+R3)+U2*(L2+R2)+U1*(L1+R1)+U0*(L0+R0);
             IF ABS(V-W)<ABS(V)*RE+AE THEN INT:=INT+ABSH*V ELSE
             BEGIN P:=P+1; PAR0[P]:=XM; PAR1[P]:=XN; PAR2[P]:=R0;
                PAR3[P]:=R1; PAR4[P]:=R2; PAR5[P]:=R3; PAR6[P]:=R4;
                PAR7[P]:=R5; PAR8[P]:=R6; PAR9[P]:=L7;
                P:=P+1; NEW:=FALSE; R0:=L7; R5:=L6; L7:=L5; L6:=L4;
                L5:=L3; L3:=L2; L2:=L1; XN:=XM;
             END
          END;
       END;
       QADRAT:=INT*16
    END;

BEGIN TIME:=CLOCK; WRITELN(' TIJD=',TIME/1000:10:3);
    MW[1]:=1; MW[2]:=3; MW[3]:=10; MW[4]:=30;
    ER[1]:=1E-3; ER[2]:=1E-6; ER[3]:=1E-9;
    PI:=4*ARCTAN(1); EVALF:=0; EVALG:=0;
    FOR SING:=1 TO 2 DO
    FOR I:=1 TO 4 DO
    FOR J:=1 TO 3 DO
    BEGIN E1[1]:=ER[J]; E1[2]:=ER[J]; E2[1]:=E1[1]/10; E2[2]:=E2[1];
```

```
        TF:=0; TG:=0; M:=MW[II];
        WRITELN(' M=',M:2,' TOL=',E1[1]:5,' INTEGRAAL=',
        QADRAT(0,1,E1,G):1:9,' F EVAL=',TF:6,' G EVAL=',TG:3);
        EVALF:=EVALF+TF; EVALG:=EVALG+TG;
        IF E1[1]=1E-9 THEN WRITELN;
        IF (E1[1]=1E-9) AND (M=30) THEN WRITELN
    END;
    WRITELN(' AANTAL EVALUATIES F:',EVALF:7);
    WRITELN(' AANTAL EVALUATIES G:',EVALG:7);
    WRITELN(' VERBRUIKTE TIJD    :',(CLOCK-TIME)/1000:9:3)
END.
```

```
"BEGIN" "COMMENT" "CHECKON" Y,Y0,DATA;
    "INTEGER" N,K,EVALDER;
    "REAL" X,XE,MU,EPS,PERIODE,TIJD;
    "REAL" "APRAY" Y,Y0[1:4],DATA[1:6];
    "BOOLEAN" FI;
    "PROCEDURE" DER(X,Y); "VALUE" X; "REAL" X; "ARRAY" Y;
    "BEGIN" "REAL" MU1,Y1,Y2,Y3,Y4,YY,N1,N2;
        MU1:=1-MU; Y1:=Y[1]; Y2:=Y[2]; Y3:=Y[3]; Y4:=Y[4];
        Y[1]:=Y3; Y[2]:=Y4; YY:=Y2*Y2;
        N1:=(Y1+MU)**2+YY; N1:=N1*SQRT(N1);
        N2:=(Y1-MU1)**2+YY; N2:=N2*SQRT(N2);
        Y[3]:=Y1+2*Y4-MU1*(Y1+MU)/N1-MU*(Y1-MU1)/N2;
        Y[4]:=Y2-2*Y3-Y2*(MU1/N1+MU/N2);
        EVALDER:=EVALDER+1;
        "IF" DATA[6]=10 "THEN" XE:=X+2*DATA[3]
    "END";

    "PROCEDURE" OUT;
    "BEGIN" "REAL" E,ERROR; "INTEGER" J;
       "IF" (X>=XE) "THEN"
       "BEGIN" ERROR:=0;
          OUTPUT(61,"("""("    X       =")",3,10D,/")",X);
          "FOR" J:=1 "STEP" 1 "UNTIL" N "DO"
          "BEGIN" E:=ABS(Y[J]-Y0[J]); "IF" E>ERROR "THEN" ERROR:=E;
             OUTPUT(61,"("ZD,"("E COMPONENT=")",-3ZD,10D,/")",J,Y[J])
          "END";
          OUTPUT(61,"("""("REJECT=")",5ZD,"(" SKIPS=")",3ZD,/,
                 "("EVALDER=")",6ZD,"(" ERROR=")",D"-ZD,
                 "(" GEVRAAGDE PRECISIE=")",D"-ZD,//")",
                 DATA[5],DATA[6],EVALDER,ERROR,EPS)
       "END"
    "END";
    "PROCEDURE" RKE(X,XE,N,Y,DER,DATA,FI,OUT);
    "VALUE" FI,N; "INTEGER" N; "REAL" X,XE; "BOOLEAN" FI;
    "ARRAY" Y,DATA; "PROCEDURE" DER,OUT;
    "BEGIN" "INTEGER" J; "BOOLEAN" LAST,FIRST,REJECT;
       "REAL" XT,H,HMIN,INT,HL,HT,ABSH,FHM,DISCR,TOL,MU,MU1,FH,E1,E2,
           C0,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11;
       "COMMENT" "CHECKON" K0,K1,K2,K3,K4;
       "ARRAY" K0,K1,K2,K3,K4[1:N];
          C0:=0,184262134833347;C1:=0,690983005625053"-1;
          C2:=0,1875;C3:=0,419262745781211;C4:=0,731762745781211;
          C5:=0,723606797749979;C6:=0,280901699437495;
          C7:=0,542705098312484;C8:=0,585410196624968;
          C9:=0,375;C10:=0,838525496156242;C11:=0,690983005625053;
       "IF" FI "THEN"
       "BEGIN" DATA[3]:=XE-X; DATA[4]:=DATA[5]:=DATA[6]:=0 "END";
       ABSH:=H:=ABS(DATA[3]);
       "IF" XE-X<0 "THEN" H:=-H; INT:=ABS(XE-X);
       HMIN:=INT*DATA[1]+DATA[2]; E1:=DATA[1]/INT; E2:=DATA[2]/INT;
       FIRST:="TRUE";
       "IF" FI "THEN" "BEGIN" LAST:="TRUE"; "GOTO" STEP "END";
   TEST; ABSH:=ABS(H); "IF" ABSH<HMIN "THEN"
       "BEGIN" H:= "IF" H>0 "THEN" HMIN "ELSE" -HMIN; ABSH:=HMIN "END";
       "IF" H>=XE-X "EQUIV" H>=0 "THEN"
       "BEGIN" LAST:="TRUE"; H:=XE-X; ABSH:=ABS(H)
       "END" "ELSE" LAST:="FALSE";
   STEP; "FOR" J:=1 "STEP" 1 "UNTIL" N "DO" K0[J]:=Y[J];
       DER(X,K0); HT:=H*C0; XT:=X+HT;
       "FOR" J:=1 "STEP" 1 "UNTIL" N "DO" K1[J]:=K0[J]*HT+Y[J];
       DER(XT,K1); HT:=H*C1; XT:=HT*4+X;
       "FOR" J:=1 "STEP" 1 "UNTIL" N "DO"K2[J]:=(K0[J]+K1[J]*3)*HT+Y[J];
       DER(XT,K2); XT:=H*0,5+X;
       "FOR" J:=1 "STEP" 1 "UNTIL" N
       "DO" K3[J]:=(K0[J]*C2-K1[J]*C3+K2[J]*C4)*H+Y[J];
       DER(XT,K3); XT:=H*C5+X;
       "FOR" J:=1 "STEP" 1 "UNTIL" N
       "DO" K4[J]:=(K0[J]*C6-K1[J]*C7+K2[J]*C8+K3[J]*0,4)*H+Y[J];
       DER(XT,K4);
       XT:="IF" LAST "THEN" XE "ELSE" X+H; HT:=H*2;
       "FOR" J:=1 "STEP" 1 "UNTIL" N "DO" K1[J]:=
       (-K0[J]*C9+K1[J]*C10-K3[J]+(K2[J]/2+K4[J])*C11)*HT+Y[J];
       DER(XT,K1);
       FHM:=0; HT:=ABSH/12;
       "FOR" J:=1 "STEP" 1 "UNTIL" N "DO"
       "BEGIN" DISCR:=ABS(K0[J]+K1[J]+K3[J]*8-(K2[J]+K4[J])*5)*HT;
            TOL:=(ABS(K0[J])*E1+E2)*ABSH;
            FH:=DISCR/TOL; "IF" FH>FHM "THEN" FHM:=FH
       "END";
       REJECT:= FHM>1; MU:=1/(1+FHM)+0,45;
```

```
        "IF" REJECT "THEN"
        "BEGIN" DATA[5]:=DATA[5]+1;
           "IF" ABSH<=HMIN "THEN"
           "BEGIN" DATA[6]:=DATA[6]+1; FIRST:="TRUE"; "GOTO" NEXT "END";
           H:=MU*H; "GOTO" TEST
        "END";
        "IF" FIRST "THEN" "BEGIN" FIRST:="FALSE"; HL:=H; H:=MU*H "END"
           "ELSE" "BEGIN" FH:=MU*H/HL+MU*MU1; HL:=H; H:=FH*H "END";
        MU1:=MU; HT:=HL/12;
        "FOR" J:=1 "STEP" 1 "UNTIL" N "DO"
        Y[J]:=(K0[J]+K1[J]+(K2[J]+K4[J])*5)*HT+Y[J];
  NEXT; DATA[3]:=HL; DATA[4]:=DATA[4]+1; X:=XT; OUT;
        "IF" X "=< XE "THEN" "GOTO" TEST
     "END";

   TIJD:=CLOCK; OUTPUT(61,"("""("TIJD=")",ZD,3D,/")",TIJD);
   Y0[1]:=1,2; Y0[2]:=0; Y0[3]:=0; Y0[4]:=1,04935750983032;
   MU:=0,0121285627653123; PERIODE:=6,19216933131963; N:=4;
   "FOR" K:=-2 "STEP" -1 "UNTIL" -9 "DO"
   "BEGIN" X:=0; XE:=PERIODE; FI:="TRUE"; EVALDER:=0;
         EPS:=EXP(K*LN(10)); DATA[1]:=EPS; DATA[2]:=EPS;
         Y[1]:=Y0[1]; Y[2]:=Y0[2]; Y[3]:=Y0[3]; Y[4]:=Y0[4];
         RKE(X,XE,N,Y,DER,DATA,FI,OUT)
   "END";
   OUTPUT(61,"("""(" VERBRUIKTE TIJD=")",ZZD,3D,/")",CLOCK-TIJD)
"END".
```

```
'BEGIN' 'CO' TEST MET PROCEDURE RKE 'CO'
    'OP' 'MAX' =('REF' 'REAL' A, 'RCAL' B) 'REAL';
         'IF' B>A 'THEN' A:=B 'FI';
    'PRIO' 'MAX' = 1;
    'PROC' RKE=('REF' 'REAL' X,XE,'INT' N, 'REF' [ ] 'REAL' Y,
        'PROC' ('REAL','REF'[ ]'REAL')'VOID' DER,'REF'[ ]'REAL' DATA,
        'BOOL' FI, 'PROC' 'VOID' OUT) 'VOID';
    'BEGIN' 'REAL' C0=0,184262134833347,C1=0,690983005625053E-1,
                   C2=0,1875,C3=0,419262745781211,C4=0,731762745781211,
                   C5=0,723606797749979,C6=0,280901699437495,
                   C7=0,542705098312484,C8=0,585410196624968,
                  C9=0,375,C10=0,838525491562421,C11=0,690983005625053;
        [1:N] 'REAL' K0,K1,K2,K3,K4;
        'IF' FI 'THEN' DATA[3]:=XE-X; DATA[4:6]:=(0,0,0) 'FI';
        'REAL' H,ABSH:=H:='ABS'(DATA[3]),INT:='ABS'(XE-X),HT,XT,HL,FHM,
             HMIN:=INT*DATA[1]+DATA[2],E1:=DATA[1]/INT,E2:=DATA[2]/INT,
             MU,MU1,'BOOL' FIRST:='TRUE',LAST:=FI,REJECT;
        'IF' XE-X<0 'THEN' H:=-H 'FI';
        'IF' FI 'THEN' 'GOTO' STEP 'FI';
  TEST: ABSH:='ABS'(H);
        'IF' ABSH<HMIN 'THEN' H:='IF' H>0 'THEN' HMIN 'ELSE' -HMIN 'FI';
             ABSH:=HMIN
        'FI';
        'IF' H>=XE-X = H>=0 'THEN' LAST:='TRUE'; H:=XE-X; ABSH:='ABS'(H)
                            'ELSE' LAST:='FALSE'
        'FI';
  STEP: K0[1:N]:=Y; DER(X,K0); HT:=H*C0; XT:=X+HT;
        'FOR' J 'TO' N 'DO' K1[J]:=Y[J]+HT*K0[J] 'OD'; DER(XT,K1);
        HT:=H*C1; XT:=X+4*HT;
        'FOR' J 'TO' N 'DO' K2[J]:=Y[J]+HT*(K0[J]+3*K1[J]) 'OD';
        DER(XT,K2); XT:=X+0,5*H;
        'FOR' J 'TO' N
        'DO' K3[J]:=Y[J]+H *(K0[J]*C2-K1[J]*C3+K2[J]*C4) 'OD';
        DER(XT,K3); XT:=X+C5*H;
        'FOR' J 'TO' N
        'DO' K4[J]:=Y[J]+H *(K0[J]*C6-K1[J]*C7+K2[J]*C8+K3[J]*0,4) 'OD';
        DER(XT,K4); XT:='IF' LAST 'THEN' XE 'ELSE' X+H 'FI'; HT:=2*H;
        'FOR' J 'TO' N 'DO' K1[J]:=Y[J]+HT*
        (-K0[J]*C9+K1[J]*C10-K3[J]+(K2[J]/2+K4[J])*C11) 'OD';
         DER(XT,K1);
        FHM:=0; HT:=ABSH/12;
        'FOR' J 'TO' N
        'DO' 'REAL'DISCR:='ABS'(K0[J]+K1[J]+K3[J]*8-(K2[J]+K4[J])*5)*HT,
             TOL:=('ABS'(K0[J])*E1+E2)*ABSH;
             'REAL' FH:=DISCR/TOL; FHM 'MAX' FH
        'OD';
        REJECT:=FHM>1; MU:=1/(1+FHM)+0,45;
        'IF' REJECT 'THEN' DATA[5]+:=1;
             'IF' ABSH<=HMIN 'THEN' DATA[6]+:=1; FIRST:='TRUE';
                  HL:=H; 'GOTO' NEXT
             'FI'; H:=MU*H; 'GOTO' TEST
        'FI';
        HL:=H;
        H:=('IF' FIRST 'THEN' MU 'ELSE' MU*H/HL+MU-MU1 'FI')*H;
        FIRST:='FALSE';
        MU1:=MU; HT:=HL/12;
        'FOR' J 'TO' N 'DO' Y[J]:=(K0[J]+K1[J]+(K2[J]+K4[J])*5)*HT 'OD';
  NEXT: DATA[3]:=HL; DATA[4]+:=1; X:=XT; OUT;
        'IF' X 'NE' XE 'THEN' 'GOTO' TEST 'FI'
    'END';

    'PROC' DEP=('REAL' X, 'REF' [ ] 'REAL' Y) 'VOID';
    'BEGIN' 'REAL' MU1=1-MU,Y1=Y[1],Y2=Y[2],Y3=Y[3],Y4=Y[4];
            Y[1:2]:=(Y3,Y4);
            'REAL' YY:=Y2**2,N1,N2;
            N1:=(Y1+MU)**2+YY; N1:=N1*SQRT(N1);
            N2:=(Y1-MU1)**2+YY; N2:=N2*SQRT(N2);
            Y[3]:=Y1+2*Y4-MU1*(Y1+MU)/N1-MU*(Y1-MU1)/N2;
            Y[4]:=Y2-2*Y3-Y2*(MU1/N1+MU/N2);
            EVALDER+:=1;
            'IF' DATA[6]=10 'THEN' XE:=X+2*DATA[3] 'FI'
    'END';

    'PROC' OUT= 'VOID';
    'BEGIN' 'IF' X>=XE 'THEN'
        'REAL' ERROR:=0;
        PRINT(("       X       =",FIXED(X,15,10),NEWLINE));
        'FOR' J 'TO' N
        'DO' 'REAL' E:='ABS'(Y[J]-Y0[J]); ERROR 'MAX' E;
          PRINT((WHOLE(J,2),"E COMPONENT=",FIXED(Y[J],15,10),NEWLINE))
```

```
        'OD';
        PRINT(("REJECT=",WHOLE(DATA[5],10)," SKIPS=",WHOLE(DATA[6],10),
          NEWLINE,"EVALDER=",WHOLE(EVALDER,7)," ERROR=",FLOAT(ERROR,9,
          2,3)," GEVRAAGDE PRECISIE=",FLOAT(DATA[1],7,2,2),NEWLINE,
          NEWLINE))
      'FI'
    'END';

    'REAL' TIJD:=CLOCK; PRINT(("TIJD=",FIXED(TIJD,10,3),NEWLINE));
   [1:4]'REAL' Y0:=(1,2,0,0,-1.0493575098303 2),Y,'REAL' X,XE,
   'BOOL' FI,'INT' EVALDER,[1:6] 'REAL' DATA;
   'INT' N=4, 'REAL' MU=0.0121285627653123, PERIODE=6.19216933131963;
   'FOR' K 'FROM' -2 'BY' -1 'TO' -9 'DO'
       X:=0; XE:=PERIODE; FI:='TRUE'; EVALDER:=0; Y[1:4]:=Y0;
       'REAL' EPS=EXP(K*LN(10));
       DATA:=(EPS,EPS,0,0,0,0);
       RKE(X,XE,4,Y,DER,DATA,FI,OUT)
    'OD';
    PRINT(("VERBRUIKTE TIJD=",FIXED(CLOCK-TIJD,10,3),NEWLINE))
'END'
```

```
      PROGRAM ORBIT(OUTPUT)
      EXTERNAL DER,OUT
      REAL X,XE,MU,EPS,PERIODE,TIJD
      INTEGER K,N,EVALDER
      REAL Y(4),Y0(4),DATA(6)
      LOGICAL FI
      COMMON /UIT/X,Y,Y0,N,EPS
      COMMON XE,MU,EVALDER,DATA
      DATA (N=4)
      TIJD=SECOND(CP) $ PRINT 5,TIJD
 5    FORMAT(* TIJD=*,F10,3)
      Y0(1)=1,2 $ Y0(2)=0, $ Y0(3)=0, $ Y0(4)=-1,04935750983032
      MU=0,012128562765312 3 $ PERIODE=6,19216933131963
      DO 100 K=2,9
      X=0, $ XE=PERIODE $ FI=,T, $ EVALDER=0 $
      EPS=EXP(-K*ALOG(10,)) $ DATA(1)=EPS $ DATA(2)=EPS
      Y(1)=Y0(1) $ Y(2)=Y0(2) $Y(3)=Y0(3) $ Y(4)=Y0(4)
      CALL RKE(X,XE,N,Y,DER,DATA,FI,OUT)
100   CONTINUE
110   FORMAT(*-VERBRUIKTE TIJD=*,F10,3)
      PRINT 110,SECOND(CP)-TIJD
      END
      SUBROUTINE DER(X,Y) $ REAL X $ DIMENSION Y(4)
      COMMON XE,MU,EVALDER,DATA $ REAL XE,MU $ DIMENSION DATA(6)
      INTEGER EVALDER
      REAL MU1,Y1,Y2,Y3,Y4,YY,N1,N2
      MU1=1-MU $ Y1=Y(1) $ Y2=Y(2) $ Y3=Y(3) $ Y4=Y(4)
      Y(1)=Y3 $ Y(2)=Y4 $ YY=Y2*Y2
      N1=Y1+MU $ N1=N1*N1+YY $ N1=N1*SQRT(N1)
      N2=Y1-MU1$ N2=N2*N2+YY $ N2=N2*SQRT(N2)
      Y(3)=Y1+2*Y4-MU1*(Y1+MU)/N1-MU*(Y1-MU1)/N2
      Y(4)=Y2-2*Y3-Y2*(MU1/N1+MU/N2) $
      EVALDER=EVALDER+1
      IF (DATA(6) ,EQ, 10,) XE=X+2*DATA(3)
      RETURN
      END
      SUBROUTINE OUT
      COMMON /UIT/X,Y,Y0,N,EPS
      REAL X,EPS $ INTEGER N $ DIMENSION Y(4),Y0(4)
      COMMON XE,MU,EVALDER,DATA $ REAL XE,MU $ DIMENSION DATA(6)
      INTEGER EVALDER
      REAL E,ERROR
      INTEGER J
      IF (X ,GE, XE) 10,20
10    ERROR=0
      PRINT 5,X
 5    FORMAT(1H0,*      X       =*,F15,10)
      DO 15 J=1,N
      E=ABS(Y(J)-Y0(J)) $ IF (E ,GT, ERROR) ERROR=E
      PRINT 11,J,Y(J)
11    FORMAT(1H ,I2,*E COMPONENT=*,F15,10)
15    CONTINUE
      PRINT 13,DATA(5),DATA(6)
13    FORMAT(* REJECT=*,F6,0,* SKIPS=*,F3,0)
      PRINT 12,EVALDER,ERROR,EPS
12    FORMAT(* EVALDER=*,I7,* ERROR=*,E10,3,* GEVRAAGDE PRECISIE=*,
     $E10,3)
20    RETURN
      END
      SUBROUTINE RKE(X,XE,N,Y,DER,DATA,FI,OUT)
      EXTERNAL DER,OUT $ REAL X,XE $ INTEGER N $ LOGICAL FI
      DIMENSION Y(4),DATA(6)
      INTEGER J
      REAL XT,H,HMIN,INT,HL,HT,ABSH,FHM,DISCR,TOL,MU,MU1,FH,E1,E2
      LOGICAL LAST,FIRST,REJECT
      REAL K0(4),K1(4),K2(4),K3(4),K4(4)
      REAL C0,C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11
      DATA(C0=0,18426213483334),(C1=0,690983005625053E-1),
     $     (C2=0,1875),(C3=0,41926274578121 1),(C4=0,73176274578121 1),
     $     (C5=0,72360677749979),(C6=0,2809016994374 95),
     $     (C7=0,54270509831248 4),(C8=0,5854101966249 68),
     $     (C9=0,375),(C10=0,83052549156242),(C11=0,690983005625053)
      IF (FI) 5,15
 5    DATA(3)=XE-X $ DATA(4)=0, $ DATA(5)=0, $ DATA(6)=0,
15    H=ABS(DATA(3)) $ ABSH=H
      IF (XE-X ,LT, 0,) H=-H $ INT=ABS(XE-X)
      HMIN=INT*DATA(1)+DATA(2) $ E1=DATA(1)/INT $ E2=DATA(2)/INT
      FIRST=,T,
      IF (FI) 20,100
20     LAST=,T, $ GOTO 200
```

```
100   ABSH=ABS(H) $ IF (ABSH .LT. HMIN) 110,120
110   IF (H .GT. 0.) 111,112
111   H=HMIN $ GOTO 113
112   H=-HMIN
113   ABSH=HMIN
120   IF ((XE-X-H)*H .LE. 0.) 130,140
130   LAST=.T. $ H=XE-X $ ABSH=ABS(H) $ GOTO 200
140   LAST=.F.
200   DO 210 J=1,N
210   K0(J)=Y(J)
      CALL DER(X,K0) $ HT=H*C0 $ XT=X+HT
      DO 220 J=1,N
220   K1(J)=K0(J)*HT+Y(J)
      CALL DER(XT,K1) $ HT=H*C1 $ XT=HT*4+X
      DO 230 J=1,N
230   K2(J)=(K0(J)+K1(J)*3)*HT+Y(J)
      CALL DER(XT,K2) $ XT=H*0.5+X
      DO 240 J=1,N
240   K3(J)=(K0(J)*C2-K1(J)*C3+K2(J)*C4)*H+Y(J)
      CALL DER(XT,K3) $ XT=H*C5+X
      DO 250 J=1,N
250   K4(J)=(K0(J)*C6-K1(J)*C7+K2(J)*C8+K3(J)*0.4)*H+Y(J)
      CALL DER(XT,K4)
      IF (LAST) 251,252
251   XT=XE $ GOTO 253
252   XT=X+H
253   HT=2*H
      DO 260 J=1,N
260   K1(J)=(-K0(J)*C9+K1(J)*C10-K3(J)+(K2(J)/2+K4(J))*C11)*HT+Y(J)
      CALL DER(XT,K1)
      FHM=0 $ HT=ABSH/12
      DO 270 J=1,N
      DISCR=ABS(K0(J)+K1(J)+K3(J)*8-(K2(J)+K4(J))*5)*HT
      TOL=(ABS(K0(J))*E1+E2)*ABSH
      FH=DISCR/TOL $ IF (FH .GT. FHM) FHM=FH
270   CONTINUE
      REJECT=(FHM .GT. 1.) $ MU=1/(1+FHM)+0.45
      IF (REJECT) 280,290
280   DATA(5)=DATA(5)+1
      IF (ABSH .LE. HMIN) 281,282
281   DATA(6)=DATA(6)+1 $ FIRST=.T. $ GOTO 400
282   H=MU*H $ GOTO 100
290   IF (FIRST) 291,292
291   FIRST=.F. $ HL=H $ H=MU*H $ GOTO 300
292   FH=MU*H/HL+MU-MU1 $ HL=H $ H=FH*H
300   MU1=MU $ HT=HL/12
      DO 310 J=1,N
310   Y(J)=(K0(J)+K1(J)+(K2(J)+K4(J))*5)*HT+Y(J)
400   DATA(3)=HL $ DATA(4)=DATA(4)+1 $ X=XT $ CALL OUT
      IF (X .NE. XE) GOTO 100
      RETURN
      END
```

```
PROGRAM ORBIT(OUTPUT);
(*$T-,$P-*)
CONST N=4;
TYPE ARN=ARRAY [1:N] OF REAL;
     AR6=ARRAY [1:6] OF REAL;
     PTR=^ARN;
VAR  X,XE,MU,EPS,PERIODE,TIJD: REAL;
     K,EVALDER: INTEGER;
     Y,Y0: ARN; DATA: AR6; FI: BOOLEAN;
PROCEDURE DER(X: REAL; Y: PTR);
    VAR MU1,Y1,Y2,Y3,Y4,YY,N1,N2: REAL;
    BEGIN MU1:=1-MU; Y1:=Y^[1]; Y2:=Y^[2]; Y3:=Y^[3]; Y4:=Y^[4];
       Y^[1]:=Y3; Y^[2]:=Y4; YY:=Y2*Y2;
       N1:=SQR(Y1+MU)+YY; N1:=N1*SQRT(N1);
       N2:=SQR(Y1-MU1)+YY; N2:=N2*SQRT(N2);
       Y^[3]:=Y1+2*Y4-MU1*(Y1+MU)/N1-MU*(Y1-MU1)/N2;
       Y^[4]:=Y2-2*Y3-Y2*(MU1/N1+MU/N2);
       EVALDER:=EVALDER+1;
       IF DATA[6]=10 THEN XE:=X+2*DATA[3]
    END;

PROCEDURE OUT;
    VAR E,ERROR: REAL; J: INTEGER;
    BEGIN IF (X>=XE) THEN
       BEGIN ERROR:=0;
          WRITELN('-      X       =',X:15:10);
          FOR J:=1 TO N DO
          BEGIN E:=ABS(Y[J]-Y0[J]); IF E>ERROR THEN ERROR:=E;
             WRITELN(J:2,'E COMPONENT=',Y[J]:15:10)
          END;
          WRITELN(' REJECT=',DATA[5]:10:0,' SKIPS=',DATA[6]:10:0);
          WRITELN(' EVALDER=',EVALDER:7,' ERROR=',ERROR:10,
                  ' GEVRAAGDE PRECISIE:',EPS:8); WRITELN
       END
    END;
PROCEDURE RKE(VAR X,XE: REAL; N: INTEGER; VAR Y: ARN; PROCEDURE DER;
              VAR DATA: AR6; FI: BOOLEAN; PROCEDURE OUT);
    LABEL 10,20,30,40;
    CONST C0=0,184262134833347;C1=0,690983005625053E-1;
          C2=0,1875;C3=0,419262745781211;C4=0,731762745781211;
          C5=0,723606797749979;C6=0,280901699437495;
          C7=0,542705098312484;C8=0,585410196624968;
          C9=0,375;C10=0,838525949156242;C11=0,690983005625053;
    VAR K0,K1,K2,K3,K4: PTR;
        J: INTEGER;
        XT,H,HMIN,INT,HL,HT,ABSH,FHM,DISCR,TOL,MU,MU1,FH,E1,E2: REAL;
        LAST,FIRST,REJECT: BOOLEAN;
    BEGIN NEW(K0); NEW(K1); NEW(K2); NEW(K3); NEW(K4);
       IF FI THEN
       BEGIN DATA[3]:=XE-X; DATA[4]:=0; DATA[5]:=0; DATA[6]:=0 END;
       H:=ABS(DATA[3]); ABSH:=H;
       IF XE-X<0 THEN H:=-H; INT:=ABS(XE-X);
       HMIN:=INT*DATA[1]+DATA[2]; E1:=DATA[1]/INT; E2:=DATA[2]/INT;
       FIRST:=TRUE;
       IF FI THEN BEGIN LAST:=TRUE; GOTO 20 END;
   10: ABSH:=ABS(H); IF ABSH<HMIN THEN
       BEGIN IF H>0 THEN H:=HMIN ELSE H:=-HMIN; ABSH:=HMIN END;
       IF (H>=XE-X) = (H>=0) THEN
       BEGIN LAST:=TRUE; H:=XE-X; ABSH:=ABS(H) END ELSE LAST:=FALSE;
   20: FOR J:=1 TO N DO K0^[J]:=Y[J]; DER(X,K0); HT:=H*C0; XT:=X+HT;
       FOR J:=1 TO N DO K1^[J]:=K0^[J]*HT+Y[J]; DER(XT,K1);
       HT:=H*C1; XT:=HT*4+X;
       FOR J:=1 TO N DO K2^[J]:=(K0^[J]+K1^[J]*3)*HT+Y[J];
       DER(XT,K2); XT:=H*0,5+X;
       FOR J:=1 TO N
       DO K3^[J]:=(K0^[J]*C2-K1^[J]*C3+K2^[J]*C4)*H+Y[J];
       DER(XT,K3); XT:=H*C5+X;
       FOR J:=1 TO N
       DO K4^[J]:=(K0^[J]*C6-K1^[J]*C7+K2^[J]*C8+K3^[J]*0,4)*H+Y[J];
       DER(XT,K4);
       IF LAST THEN XT:=XE ELSE XT:=X+H; HT:=H*2;
       FOR J:=1 TO N DO K1^[J]:=
       (-K0^[J]*C9+K1^[J]*C10-K3^[J]+(K2^[J]/2+K4^[J])*C11)*HT+Y[J];
       DER(XT,K1);
       FHM:=0; HT:=ABSH/12;
       FOR J:=1 TO N DO
       BEGIN DISCR:=ABS(K0^[J]+K1^[J]+K3^[J]*8-(K2^[J]+K4^[J])*5)*HT;
             TOL:=(ABS(K0^[J])*E1+E2)*ABSH;
             FH:=DISCR/TOL; IF FH>FHM THEN FHM:=FH
       END;
```

```
        REJECT:= FHM>1; MU:=1/(1+FHM)+0.45;
        IF REJECT THEN
        BEGIN DATA[5]:=DATA[5]+1;
           IF ABSH<=HMIN THEN
           BEGIN DATA[6]:=DATA[6]+1; FIRST:=TRUE; GOTO 40 END;
           H:=MU*H; GOTO 10
        END;
        IF FIRST THEN BEGIN FIRST:=FALSE; HL:=H; H:=MU*H END ELSE
        BEGIN FH:=MU*H/HL+MU-MU1; HL:=H; H:=FH*H END;
        MU1:=MU; HT:=HL/12;
        FOR J:=1 TO N DO Y[J]:=(K0^[J]+K1^[J]+(K2^[J]+K4^[J])*5)*HT+Y[J];
   40:  DATA[3]:=HL; DATA[4]:=DATA[4]+1; X:=XT; OUT;
        IF X <> XE THEN GOTO 10;
        DISPOSE(K0); DISPOSE(K1); DISPOSE(K2); DISPOSE(K3); DISPOSE(K4)
    END;
BEGIN TIJD:=CLOCK; WRITELN('1TIJD=',TIJD/1000:10:3);
   Y0[1]:=1.2; Y0[2]:=0; Y0[3]:=0; Y0[4]:=-1.04935750983032;
   MU:=0.012128562765312 3; PERIODE:=6.19216933131963;
   FOR K:=-2 DOWNTO -9 DO
   BEGIN X:=0; XE:=PERIODE; FI:=TRUE; EVALDER:=0;
         EPS:=EXP(K*LN(10)); DATA[1]:=EPS; DATA[2]:=EPS;
         Y[1]:=Y0[1]; Y[2]:=Y0[2]; Y[3]:=Y0[3]; Y[4]:=Y0[4];
         RKE(X,XE,N,Y,DER,DATA,FI,OUT)
   END;
   WRITELN(' VERBRUIKTE TIJD=',(CLOCK-TIJD)/1000:10:3)
END.
```

```
"BEGIN" "COMMENT" CHOLESKY DECOMPOSITION ON CHANGED HILBERT-MATRICES;
    "REAL" "PROCEDURE" VECVEC(L, U, SHIFT, A, B); "VALUE" L,U,SHIFT;
    "INTEGER" L,U,SHIFT; "ARRAY" A,B;
    "BEGIN" "INTEGER" K; "REAL" S;
        S:= 0;
        "FOR" K:=L "STEP" 1 "UNTIL" U "DO" S:= A[K] * B[SHIFT + K] + S;
        VECVEC:= S
    "END" VECVEC;

    "REAL" "PROCEDURE" SEQVEC(L, U, IL, SHIFT, A, B);
    "VALUE" L,U,IL,SHIFT; "INTEGER" L,U,IL,SHIFT; "ARRAY" A,B;
    "BEGIN" "INTEGER" K; "REAL" S;
        S:= 0;
        "FOR" L:=L "STEP" 1 "UNTIL" U "DO"
        "BEGIN" S:= A[IL] * B[L + SHIFT] + S; IL:= IL + L "END";
        SEQVEC:= S
    "END" SEQVEC;

    "REAL" "PROCEDURE" SYMMATVEC(L, U, I, A, B); "VALUE" L,U,I;
    "INTEGER" L,U,I; "ARRAY" A,B;
    "BEGIN" "INTEGER" K, M;
        M:= "IF" L > I "THEN" L "ELSE" I; K:= M * (M - 1) // 2;
        SYMMATVEC:= VECVEC(L, "IF" I <= U "THEN" I-1 "ELSE" U, K, B, A)
                  + SEQVEC(M, U, K + I, 0, A, B)
    "END" SYMMATVEC;

     "PROCEDURE" CHLDEC1(A, N, AUX); "VALUE" N; "INTEGER" N;
     "ARRAY" A, AUX;
     "BEGIN" "INTEGER" I, J, K, KK, KJ, LOW, UP; "REAL" R, EPSNORM;
         R:= 0; KK:= 0;
         "FOR" K:= 1 "STEP" 1 "UNTIL" N "DO"
         "BEGIN" KK:= KK + K; "IF" A[KK] > R "THEN" R:= A[KK] "END";
         EPSNORM:= AUX[2] * R; KK:= 0;
         "FOR" K:= 1 "STEP" 1 "UNTIL" N "DO"
         "BEGIN" KK:= KK + K; LOW:= KK - K + 1; UP:= KK - 1;
             R:= A[KK] - VECVEC(LOW, UP, 0, A, A);
             "IF" R <= EPSNORM "THEN"
             "BEGIN" AUX[3]:= K - 1; "GOTO" END "END";
             A[KK]:= R:= SQRT(R); KJ:= KK + K;
             "FOR" J:= K + 1 "STEP" 1 "UNTIL" N "DO"
             "BEGIN" A[KJ]:= (A[KJ] -
                     VECVEC(LOW, UP, KJ - KK, A, A)) / R;
                 KJ:= KJ + J
             "END"
         "END";
         AUX[3]:= N;
     END;
     "END" CHLDEC1;

     "PROCEDURE" CHLSOL1(A, N, B); "VALUE" N; "INTEGER" N; "ARRAY" A, B;
     "BEGIN" "INTEGER" I, II;
         II:= 0;
         "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
         "BEGIN" II:= II + I;
             B[I]:= (B[I] - VECVEC(1, I - 1, II - I, B, A)) / A[II]
         "END";
         "FOR" I:= N "STEP" -1 "UNTIL" 1 "DO"
         "BEGIN" B[I]:= (B[I] -
                 SEQVEC(I + 1, N, II + I, 0, A, B)) / A[II];
             II:= II - I
         "END"
     "END" CHLSOL1;

     "PROCEDURE" CHLDECSOL1(A, N, AUX, B); "VALUE" N; "INTEGER" N;
     "ARRAY" A, AUX, B;
     "BEGIN"
         CHLDEC1(A, N, AUX);
         "IF" AUX[3] = N "THEN" CHLSOL1(A, N, B)
     "END" CHLDECSOL1;


    "INTEGER" N;
    "REAL" TOTALCLOCK, BUILDUPCLOCK, CHOLCLOCK;
    TOTALCLOCK:= CLOCK; BUILDUPCLOCK:= CHOLCLOCK:= 0;
    "FOR" N:= 10 "STEP" 30 "UNTIL" 100 "DO"
    "BEGIN" "INTEGER" I, J, IJ;
        "ARRAY" A[1 : (N + 1) * N // 2], B, X[1 : N], AUX[2 : 3];
        IJ:= 0; AUX[2]:= -1;
        BUILDUPCLOCK:= BUILDUPCLOCK - CLOCK;
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
```

```
        "BEGIN" "FOR" J:= 1 "STEP" 1 "UNTIL" I - 1 "DO"
           "BEGIN" IJ:= IJ + 1; A[IJ]:= 1 / (I + J - 1) "END";
             IJ:= IJ + 1; A[IJ]:= 0,1 + 1 / (I + J - 1)
        "END";
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO" X[I]:= I;
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        B[I]:= SYMMATVEC(1, N, I, A, X);
        BUILDUPCLOCK:= BUILDUPCLOCK + CLOCK;
        OUTPUT(61, "("2/,"("GIVEN SOLUTION VECTOR AND RIGHT HAND SIDE")",
        /")");
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        OUTPUT(61, "("+2ZD,5D3B")", X[I]);
        OUTPUT(61,"("/")");
        "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
        OUTPUT(61, "("+2ZD,5D3B")", B[I]);
        OUTPUT(61,"("2/")");
        CHOLCLOCK:= CHOLCLOCK - CLOCK;
        CHLDECSOL1(A, N, AUX, B);
        CHOLCLOCK:= CHOLCLOCK + CLOCK;
        "IF" AUX[3] "=" N "THEN"
        OUTPUT(61, "("""("BREAKDOWN AT")", -6ZD, /")", AUX[3] + 1)
        "ELSE"
        "BEGIN" OUTPUT(61, "("""("CALCULATED SOLUTION VECTOR")", /")");
            "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO" OUTPUT(61,"("N")", B[I]);
            OUTPUT(61,"("/")")
        "END"
    "END";
    TOTALCLOCK:= CLOCK - TOTALCLOCK;
    OUTPUT(61, "("*, "("TIJDEN")", 2/,
       "("OPBOUW GEGEVENS: ")", N, /, "("OPLOSSEN: ")", N, /,
       "("AFDRUKKEN: ")", N, /, "("TOTAAL: ")", N")",
       BUILDUPCLOCK, CHOLCLOCK, TOTALCLOCK - BUILDUPCLOCK - CHOLCLOCK,
       TOTALCLOCK)
"END"
```

```
'BEGIN' #CHOLESKY DECOMPOSITION PERFORMED ON SEGMENTS OF HILBERT'S
   MATRIX + 0,1 ON DIAG, (WITH SOME IDEAS FROM VAN DER MEULEN'S TORRIX),
    TJDEKKER 750320 #
'REAL' ARREB = 1,0E-11;
'MODE' 'VECTOR' = 'REF'[]'REAL';
'MODE' 'MULVEC' = 'STRUCT'('REAL' FAC, 'VECTOR' VEC);
'MODE' 'VECROW' = 'REF'[]'VECTOR';
'MODE' 'ERRIND' = 'STRUCT'('BOOL' OKAY, 'INT' FAILAT);

'OP' 'MAX' = ('INT' A, B)'INT'; 'IF' A < B 'THEN' B 'ELSE' A 'FI';
'PRIO' 'MAX' = 9;

'OP' 'MIN' = ('INT' A, B)'INT'; 'IF' A < B 'THEN' A 'ELSE' B 'FI';
'PRIO' 'MIN' = 9;

'OP' * = ('VECTOR' A, B)'REAL'; # SCALAR PRODUCT #
'BEGIN' 'REAL' S:= 0;
    'FOR' K 'FROM' 'LWB' A 'MAX' 'LWB' B 'TO' 'UPB' A 'MIN' 'UPB' B
    'DO' S+:= A[K] * B[K] 'OD';  S
'END';

'OP' * = ('REAL' R, 'VECTOR' A)'MULVEC'; 'MULVEC'(R, A);

'OP' -:= = ('VECTOR' A, 'MULVEC' MV)'VECTOR';
  # ELIMINATION STEP; VECTOR MINUS SCALAR TIMES VECTOR #
'BEGIN' 'REAL' X = FAC 'OF' MV, 'VECTOR' V = VEC 'OF' MV;
    'FOR' K 'FROM' 'LWB' A 'MAX' 'LWB' V 'TO' 'UPB' A 'MIN' 'UPB' V
    'DO' A[K]-:= X * V[K] 'OD';  A
'END';

'OP' +:= = ('VECTOR' U, 'MULVEC' MV)'VECTOR';
  # VECTOR PLUS SCALAR TIMES VECTOR #
'BEGIN' 'REAL' X = FAC 'OF' MV, 'VECTOR' V = VEC 'OF' MV;
    'FOR' K 'FROM' 'LWB' U 'MAX' 'LWB' V 'TO' 'UPB' U 'MIN' 'UPB' V
    'DO' U[K]+:= X * V[K] 'OD';  U
'END';

'OP' * = ('VECROW' A, 'VECTOR' U)'VECTOR';
'BEGIN' 'INT' LOW = 'LWB' A,UP = 'UPB' A;
    'VECTOR' PROD = 'HEAP'[LOW : UP]'REAL';
    'FOR' K 'FROM' LOW 'TO' UP 'DO' PROD[K]:= A[K] * U 'OD';
    PROD
'END';

'OP' 'SYMMATVEC' = ('VECROW' A, 'VECTOR' U)'VECTOR';
  # SYMMETRIC MATRIX TIMES VECTOR #
'BEGIN' 'VECTOR' PROD = A * U; 'INT' LOW = 'LWB' PROD;
    'FOR' K 'FROM' 'LWB' A 'MAX' 'LWB' U 'TO' 'UPB' A 'MIN' 'UPB' U
    'DO' PROD[ : K - 1 @ LOW]+:= U[K] * A[K] 'OD';
    PROD
'END';
'PRIO' 'SYMMATVEC' = 9;

'OP' 'PRIVEC' = ('VECTOR' U)'VOID';
'BEGIN' 'INT' LOW = 'LWB' U;
    'FOR' K 'TO' LOW - 1 'DO' PRINT(0) 'OD';
    'FOR' K 'FROM' LOW 'TO' 'UPB' U 'DO' PRINT(U[K]) 'OD';
    PRINT(NEWLINE)
'END';

'OP' 'PRIROWVEC' = ('VECROW' A)'VOID';
'BEGIN' 'FOR' K 'FROM' 'LWB' A 'TO' 'UPB' A 'DO' 'PRIVEC' A[K] 'OD';
    PRINT(NEWLINE)
'END';

'PROC' CHODEC = ('VECROW' A, 'REF''ERRIND' IND)'VOID';
  # CHOLESKY DECOMPOSITION
    IN A ONE SHOULD GIVE THE LOWER-TRIANGULAR PART OF THE MATRIX,
    VIZ, EACH ROW FROM ITS FIRST NON-ZERO ELEMENT TO ITS DIAGONAL
    ELEMENT; CHOLESKY DECOMPOSITION PRODUCES A MATRIX OF THE SAME SHAPE#
'BEGIN' 'INT' N = 'UPB' A;
    'REAL' R:= 0;
    'FOR' K 'TO' N
    'DO' 'REAL' AKK = A[K][K]; 'IF' AKK > R 'THEN' R:= AKK 'FI' 'OD';
    'REAL' EPSNORM:= ARREB * R;
    'FOR' K 'TO' N
    'DO' 'VECTOR' AK = A[K]; 'INT' LOW = 'LWB' AK;
         'FOR' J 'FROM' 'LWB' AK 'TO' K - 1
         'DO' (AK[J]-:= AK[ : J - 1 @ LOW] * A[J])/:= A[J][J] 'OD';
         'REF''REAL' RAKK = A[K][K];
```

```
        'REAL' D := RAKK - AK[ : K - 1 @ LOW] * AK;
        'IF' D <= EPSNORM
        'THEN' OKAY 'OF' IND:= 'FALSE'; FAILAT 'OF' IND:= K; 'GOTO' EX
        'FI';
        RAKK:= SQRT(D)
    'OD';
    OKAY 'OF' IND:= 'TRUE';
EX: 'SKIP'
'END' # CHODEC # ;

'PROC' CHOSOL = ('VECROW' A, 'VECTOR' B)'VOID';
  # FORWARD AND BACK SUBSTITUTION FOR CHOLESKY DECOMPOSED MATRIX #
'BEGIN' 'INT' N = 'UPB' A, LOW = 'LWB' B;
    'FOR' J 'TO' N
    'DO' (B[J]-:= B[ : J - 1 @ LOW] * A[J])/:= A[J][J] 'OD';
    'FOR' K 'FROM' N 'BY' -1 'TO' 1
    'DO' 'REF''REAL' RXK = B[K];
        RXK/:= A[K][K];
        B[ : K - 1 @ LOW]-:= RXK * A[K]
    'OD'
'END' # CHOSOL # ;

'PROC' CHODECSOL = ('VECROW' A, 'VECTOR' B, 'REF' 'ERRIND' IND)'VOID';
  # SOLVES SYMMETRIC POSITIVE DEFINITE SYSTEM #
'BEGIN' CHODEC(A, IND);
    'IF' OKAY 'OF' IND 'THEN' CHOSOL(A, B) 'FI'
'END';

'BEGIN'  # MAIN PROGRAM #
    'REAL' TOTALCLOCK:= CLOCK, BUILDUPCLOCK:= 0, CHOLCLOCK:= 0;
    'FOR' N 'FROM' 10 'BY' 30 'TO' 100
    'DO' 'VECTOR' X = 'LOC'[1 : N]'REAL', B = 'LOC'[1 : N]'REAL',
        'VECROW' A = 'LOC'[1 : N]'VECTOR',
        'REF''ERRIND' IND = 'LOC''ERRIND';
        BUILDUPCLOCK-:= CLOCK;
        'FOR' I 'TO' N
        'DO' 'VECTOR' AI = 'HEAP'[1 : I]'REAL';
            'FOR' J 'TO' I - 1 'DO' AI[J]:= 1 / (I + J - 1) 'OD';
            AI[I]:= 0,1 + 1 /.(I + I - 1);
            A[I]:= AI; X[I]:= I
        'OD';
        B:= A 'SYMMATVEC' X;
        BUILDUPCLOCK+:= CLOCK;
        PRINT((NEWLINE, NEWLINE));
        PRINT(("GIVEN SOLUTION VECTOR AND RIGHT HAND SIDE", NEWLINE));
        'PRIVEC' X; 'PRIVEC' B; PRINT(NEWLINE);
        CHOLCLOCK-:= CLOCK;
        CHODECSOL(A, B, IND);
        CHOLCLOCK+:= CLOCK;
        'IF' OKAY 'OF' IND
        'THEN' PRINT(("CALCULATED SOLUTION VECTOR", NEWLINE));
            'PRIVEC' B
        'ELSE' PRINT(("BREAKDOWN AT", FAILAT 'OF' IND,NEWLINE))
        'FI'
    'OD';
    TOTALCLOCK:= CLOCK - TOTALCLOCK;
    PRINT('NEWPAGE, "TIJDEN", NEWLINE, NEWLINE,
       "OPBOUW: ", BUILDUPCLOCK, NEWLINE, "OPLOSSEN: ", CHOLCLOCK,
       NEWLINE, "UITVOER: ", TOTALCLOCK - CHOLCLOCK - BUILDUPCLOCK,
       NEWLINE, "TOTAAL: ", TOTALCLOCK))
'END'
'END'
```

```
PROGRAM CHOLESKY(OUTPUT);
(* CHOLESKY DECOMPOSITION FOR SEGMENTS OF HILBERT'S MATRIX + 0,1 ON DIAG
   TJOEKKER 750317, CHODECSOL IS EQUIVALENT TO CHLDECSOL1 FROM
   NUMAL3 SECTION 3,1,1,1,1,2,3 *)
    CONST RELPR = 1,0E-11;
          SIZ = 100;
          TALSIZ = 5050; (* (SIZ + 1) * SIZ DIV 2 *)
    TYPE  INX = 1 .. SIZ;
          TALINX = 1 .. TALSIZ;
          VEC = ARRAY[ INX ] OF REAL;
          TAL = ARRAY[ TALINX ] OF REAL;
          (* THE UPPER TRIANGLE OF THE SYMMETRIC POSITIVE DEFINITE
             MATRIX MUST BE GIVEN IN TAL; THE (I,J)-TH ELEMENT  OF
             THE MATRIX MUST BE GIVEN IN TAL[(J - 1) * J DIV 2 + I]
             FOR 1 <= I <= J <= SIZ *)
          ERRIND = RECORD CASE OKAY : BOOLEAN OF
                              TRUE : ();
                              FALSE : (FALINX : INX)
                   END;

    FUNCTION TALVEC(LOW, UP : INTEGER; I : INX; VAR A : TAL;
                    VAR B : VEC) : REAL;
    (* THE I-TH ROW OF THE UPPER-TRIANGLE TIMES VECTOR *)
        VAR S : REAL;
            K : INX;
            IK : INTEGER;
    BEGIN (* TALVEC *)
        S := 0; IK := (LOW - 1) * LOW DIV 2 + I;
        FOR K := LOW TO UP DO
        BEGIN S:= A[IK] * B[K] + S;
            IK := IK + K (* IK = (K - 1) * K DIV 2 + I *)
        END;
        TALVEC := S
    END; (* TALVEC *)

    FUNCTION LATVEC(LOW, UP : INTEGER; J : INX; VAR A : TAL;
                    VAR B : VEC) : REAL;
    (* THE J-TH COLUMN OF THE UPPER-TRIANGLE TIMES VECTOR *)
        VAR S : REAL;
            K : INX;
            JO : INTEGER;
    BEGIN (* LATVEC *)
        S := 0; JO := (J - 1) * J DIV 2;
        FOR K := LOW TO UP DO S := A[JO + K] * B[K] + S;
        LATVEC := S
    END; (* LATVEC *)

    FUNCTION SYMVEC(N, I : INX; VAR A : TAL; VAR B : VEC) : REAL;
    (* THE I-TH ROW OF SYMMETRIC MATRIX TIMES VECTOR *)
    BEGIN SYMVEC := LATVEC(1,I,I,A,B) + TALVEC(I + 1,N,I,A,B)
    END; (* SYMVEC *)

    FUNCTION LATTAL(LOW, UP : INTEGER; I, J : INX; VAR A : TAL) : REAL;
    (* I-TH TIMES J-TH COLUMN OF UPPER TRIANGLE *)
        VAR S : REAL;
            IO, SHIFT, K : INTEGER;
    BEGIN (* LATTAL *)
        S := 0; IO := (I -1) * I DIV 2; SHIFT := (J - 1) * J DIV 2 - IO;
        FOR K := IO + LOW TO IO + UP DO S := A[K] * A[SHIFT + K] + S;
        LATTAL := S
    END; (* LATTAL *)

    PROCEDURE CHODEC(N: INX; VAR A :TAL; VAR IND : ERRIND);
    (* CHOLESKY DECOMPOSITION *)
        LABEL 1;
        VAR K, J : INX;
            KK, KJ : TALINX;
            EPSNORM, R : REAL;
    BEGIN(* CHODEC *)
        KK := 1; R := 0;
        FOR K := 1 TO N DO
        BEGIN IF K > 1 THEN KK := KK + K;
            IF A[KK] > R THEN R := A[KK]
        END;
        EPSNORM := RELPR * R; KK := 1;
        FOR K := 1 TO N DO
        BEGIN IF K > 1 THEN KK := KK + K;
            R := A[KK] - LATTAL(1, K - 1, K, K, A);
            IF R <= EPSNORM THEN
            BEGIN IND.OKAY := FALSE; IND.FALINX := K; GOTO 1 END;
```

```
            R := SQRT(R); A[KK] := R;
            IF K < N THEN KJ := KK + K;
            FOR J := K + 1 TO N DO
            BEGIN A[KJ] := ( A[KJ] - LATTAL(1, K - 1, K, J, A) ) / R;
                IF J < N THEN KJ := KJ + J
            END;
        END;
        IND.OKAY := TRUE;
    1 :
    END; (* CHODEC *)

    PROCEDURE CHOSOL(N : INX; VAR A: TAL; VAR B : VEC);
    (* FORWARD AND BACK SUBSTITUTION FOR CHOLESKY DECOMPOSED MATRIX *)
        VAR K : INX;
            KK : INTEGER;
    BEGIN (* CHOSOL *)
        KK := 0;
        FOR K := 1 TO N DO
        BEGIN KK := KK + K;
            B[K] := (B[K] - LATVEC(1, K - 1, K, A, B)) / A[KK]
        END;
        FOR K := N DOWNTO 1 DO
        BEGIN B[K] := (B[K] - TALVEC(K + 1, N, K, A, B)) / A[KK];
            KK := KK - K
        END
    END; (* CHOSOL *)

    PROCEDURE CHODECSOL(N : INX; VAR A : TAL; VAR B : VEC;
                        VAR IND : ERRIND);
    (* SOLVES SYMMETRIC POSITIVE DEFINITE SYSTEM BY CALLING CHODEC AND,
       IF THIS CALL WAS SUCCESFUL, CHOSOL  *)
    BEGIN (* CHODECSOL *)
        CHODEC(N, A, IND);
        IF IND.OKAY THEN CHOSOL(N, A, B)
    END; (* CHODECSOL *)

    PROCEDURE MAIN1;
       VAR I, J : INX;
           N, IJ: INTEGER;
           A : TAL;
           B, X : VEC;
           IND : ERRIND;
           TOTALCLOCK, BUILDUPCLOCK, CHOLCLOCK: REAL;
    BEGIN TOTALCLOCK:= CLOCK; BUILDUPCLOCK:= 0; CHOLCLOCK:= 0;
       N:= -20;
       REPEAT N:= N + 30;
          IJ:= 0;
          BUILDUPCLOCK:= BUILDUPCLOCK - CLOCK;
          FOR I:= 1 TO N DO
          BEGIN FOR J:= 1 TO I - 1 DO
             BEGIN IJ:= IJ + 1; A[IJ]:= 1 / (I + J - 1) END;
             IJ:= IJ + 1; A[IJ]:= 0.1 + 1 / (I + I - 1)
          END;
          FOR I:= 1 TO N DO X[I]:= I;
          FOR I:= 1 TO N DO B[I]:= SYMVEC(N, I, A, X);
          BUILDUPCLOCK:= BUILDUPCLOCK + CLOCK; WRITELN; WRITELN;
          WRITELN('0GIVEN SOLUTION VECTOR AND RIGHT HAND SIDE');
          FOR I:= 1 TO N DO WRITE(X[I] : 12 : 5); WRITELN;
          FOR I:= 1 TO N DO WRITE(B[I] : 12 : 5); WRITELN;
          CHOLCLOCK:= CHOLCLOCK - CLOCK;
          CHODECSOL(N, A, B, IND);
          CHOLCLOCK:= CHOLCLOCK + CLOCK;
          IF IND.OKAY THEN
          BEGIN WRITELN; WRITELN('0CALCULATED SOLUTION VECTOR');
             FOR I:= 1 TO N DO WRITE(B[I] : 20); WRITELN
          END ELSE
          BEGIN WRITELN; WRITELN('0BREAKDOWN AT ', IND.FALINX) END;
       UNTIL N = 100;
       TOTALCLOCK:= CLOCK - TOTALCLOCK; WRITE('1TIJDEN:');  WRITELN;
       WRITELN; WRITELN(' OPBOUW: ', BUILDUPCLOCK / 1000);
       WRITELN(' OPLOSSEN: ', CHOLCLOCK / 1000);
       WRITELN(' UITVOER: ', (TOTALCLOCK - BUILDUPCLOCK - CHOLCLOCK)
          / 1000);
       WRITE(' TOTAAL: ', TOTALCLOCK / 1000)
    END (* MAIN1 *);

BEGIN (* CHOLESKY *)
    MAIN1
END .
```

```
"BEGIN"
"PROCEDURE" TRED1(N,TOL) TRANS:(A) RESULT:(D,E,E2);
"VALUE" N,TOL; "INTEGER" N; "REAL" TOL; "ARRAY" A,D,E,E2;
  "COMMENT" "CHECKON"      A,D,E,E2;
"COMMENT" THIS PROCEDURE REDUCES THE GIVEN LOWER TRIANGLE OF A SYMMETRIC
  MATRIX, A, STORED IN THE ARRAY A[1:N,1:N], TO TRIDIAGONAL FORM USING
  HOUSEHOLDERS REDUCTION, THE DIAGONAL OF THE RESULT IS STORED IN THE
  ARRAY D[1:N] AND THE SUB-DIAGONAL IN THE LAST N-1 STORES OF THE ARRAY
  E[1:N] (WITH THE ADDITIONAL ELEMENT E[1] = 0), E2[I] IS SET TO EQUAL
  E[I] ** 2, THE STRICTLY LOWER TRIANGLE OF THE ARRAY A, TOGETHER WITH
  THE ARRAY E,IS USED TO STORE SUFFICIENT INFORMATION FOR THE DETAILS OF
  THE TRANSFORMATION TO BE RECOVERABLE IN THE PROCEDURE TRBAK1, THE
  UPPER TRIANGLE OF THE ARRAY IS LEFT UNALTERED;
"BEGIN" "INTEGER" I,J,K,L;
  "REAL" F,G,H;
  "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
  D[I]:= A[I,I];
  "FOR" I:= N "STEP" -1 "UNTIL" 1 "DO"
  "BEGIN" L:= I-1; H:= 0;
    "FOR" K:= 1 "STEP" 1 "UNTIL" L "DO"
    H:= H + A[I,K] * A[I,K];
    "COMMENT" IF H IS TOO SMALL FOR ORTHOGONALITY TO BE GUARANTEED, THE
      TRANSFORMATION IS SKIPPED;
    "IF" H <= TOL "THEN"
    "BEGIN" E[I]:= E2[I]:= 0; "GO TO" SKIP
    "END";
    E2[I]:= H; F:= A[I,I-1];
    E[I]:= G:= "IF" F >= 0 "THEN" -SQRT(H) "ELSE" SQRT(H);
    H:= H - F*G; A[I,I-1]:= F - G; F:= 0;
    "FOR" J:= 1 "STEP" 1 "UNTIL" L "DO"
    "BEGIN" G:= 0;
      "COMMENT" FORM ELEMENT OF A*U;
      "FOR" K:= 1 "STEP" 1 "UNTIL" J "DO"
      G:= G + A[J,K] * A[I,K];
      "FOR" K:= J+1 "STEP" 1 "UNTIL" L "DO"
      G:= G+A[K,J] * A[I,K];
      "COMMENT" FORM ELEMENT OF P;
      G:= E[J]:= G/H; F:= F + G * A[I,J]
    "END" J;
    "COMMENT" FORM K;
    H:= F/(H+H);
    "COMMENT" FORM REDUCED A;
    "FOR" J:= 1 "STEP" 1 "UNTIL" L "DO"
    "BEGIN" F:= A[I,J]; G:= E[J]:= E[J] - H * F;
      "FOR" K:= 1 "STEP" 1 "UNTIL" J "DO"
      A[J,K]:= A[J,K] - F * E[K] - G * A[I,K]
    "END" J;
  SKIP: H:= D[I]; D[I]:= A[I,I]; A[I,I]:= H
  "END" I
"END" TRED1;
"INTEGER" I,J,K,N; "REAL" TOL;

   N:=25;
OUTPUT(61,"("""(" DIMENSIE MATRIX")",N")",N);
    OUTPUT(61,"("/")");
"BEGIN"
"ARRAY" COPYA[1:N,1:N]; "INTEGER" ITELL;
"REAL""ARRAY" A[1:N,1:N],D,E,E2[1:N];
"COMMENT" CHECKON" A,COPYA,D,E,E2;
"REAL"TIME;
"FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
"FOR" J:=1 "STEP" 1 "UNTIL" N "DO"
A[I,J]:=COPYA[I,J]:="IF" I>J "THEN" I "ELSE" J;
TOL:="-200;
TIME:=CLOCK;
"FOR" ITELL:=1 "STEP" 1 "UNTIL" 10  "DO"  "BEGIN"
TRED1(N,TOL,A,D,E,E2);
"FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
"FOR" J:=1 "STEP" 1 "UNTIL" N "DO"
A[I,J]:=COPYA[I,J];
"END" TESTCYCLUS;
TIME:=CLOCK-TIME;

OUTPUT(61,"("///")");
OUTPUT(61,"("""(" DE DIAGONAAL")""")");
OUTPUT(61,"("/")");
"FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
OUTREAL(61,D[I]);
OUTPUT(61,"("/")");
OUTPUT(61,"("""(" EN DE NEVENDIAGONAAL")""")");
```

```
"FOR" I:=1 "STEP" 1 "UNTIL" N "DO"
 OUTREAL(61,E[I]);
OUTPUT(61,"("("TOTAALTIJD VAN DE TESTCYCLUS:")",N")",TIME);
"END";
"END";
```

```
(
'PROC' TRED1 =('INT' N,'REAL' TOL,'REF'[ , ]'REAL' A,'REF'[ ]'REAL'
                        D,E,E2)'VOID';

   'CO' TRIDIAGONALISATION OF SYMMETRIC MATRIX
       TRANSLATION OF PROCEDURE TRED1 DIRIVED FROM WILKINSON AND REINSCH
        LINEAR ALGEBRA HANDBOOK
   'CO'

  (
  'INT' L,'REAL' F,G,H;
  'FOR' I 'TO' N 'DO' D[I]:=A[I,I] 'OD';
  'FOR' I 'FROM' N 'BY' -1 'TO'1'DO'
    L:=I-1;
    H:=0;
    'FOR'K'TO'L'DO'H+:=A[I,K]*A[I,K] 'OD';
   'IF' H<=TOL 'THEN'
   E[I]:=E2[I]:=0; 'GOTO' SKIP 'FI';
   E2[I]:=H; F:=A[I,I-1];
  E[I]:=G:='IF' F>=0 'THEN' -SQRT(H) 'ELSE' SQRT(H) 'FI';
  H-:=F*G; A[I,I-1]:=F-G; F:=0;
  'FOR' J 'TO' L 'DO'
     G:=0;
    'FOR'K'TO'J'DO'G+:=A[J,K]*A[I,K] 'OD';
    'FOR'K'FROM'J+1'TO'L'DO' G+:=A[K,J]*A[I,K] 'OD';
   F+:=(E[J]:=G/H)*A[I,J]
   'OD';
  H:=F/(H+H);
  'FOR' J 'TO' L 'DO'
     G:=E[J]-:=H*(F:=A[I,J]);
    'FOR'K'TO'J'DO' A[J,K]-:=F*E[K]+G*A[I,K] 'OD'
   'OD';

 SKIP;
  H:=D[I]; D[I]:=A[I,I]; A[I,I]:=H
   'OD'
  );
'INT' N:=25,J,K,'REAL' TOL:=1.0E-200;
 [1:N,1:N]'REAL'A,B,[1:N]'REAL'D,E,E2;
[]'CHAR'HEADING="TRED1 IN ALGOL68**********************************";
   'FOR' J 'TO' N 'DO'
   'FOR' K 'TO' N 'DO'
   B[J,K]:=(J>K|J|K)
'OD'  'OD';
   PRINT((NEWLINE,HEADING,NEWLINE,NEWLINE," DIMENSIE MATRIX  : ",N));
'REAL'T1:=CLOCK;
  'TO' 10 'DO'
A:=B;
  TRED1(N,TOL,A,D,E,E2)
'OD';T1:=CLOCK-T1;
   PRINT((NEWLINE,NEWLINE,"HOOFD-DIAGONAAL:",NEWLINE,D,
         NEWLINE,"NEVEN-DIAGONAAL:",NEWLINE,E,NEWLINE,
         "REKENTIJD: ",T1," SECONDE'I"))
)
```

```
      PROGRAM TREDTE(INPUT,OUTPUT)
        DIMENSION A(25,25),COPYA(25,25),E(25),D(25),E2(25)
           N=25
        DO 1I=1,N
        DO 1J=1,N
1         A(I,J)=COPYA(I,J)=FLOAT(MAXO(I,J))
        PRINT 100,N
100     FORMAT(*1 DIMENSIE MATRIX: *,I10,//)
        TIME=SECOND(TTTT)
           DO 202 II=1,10
        DO 20I=1,N
         DO 20J=1,N
20      A(I,J)=COPYA(I,J)
         CALL TRED1(N,N,A,D,E,E2)
202     CONTINUE
        TIME=SECOND(TTTT)-TIME
        PRINT 1060,D
1060    FORMAT(///,* DE DIAGONAAL: *,5(F15.8,*  *))
        PRINT 1072,E
1072    FORMAT(///,* EN DE NEVENDIAGONAAL:  *,5(F15.8,*  *))
        PRINT 2000,TIME
2000    FORMAT(///,* CP-TIME :  *,F8.3)
        END
      SUBROUTINE TRED1(NM,N,A,D,E,E2)
C
      INTEGER I,J,K,L,N,II,NM,JP1
      REAL A(NM,N),D(N),E(N),E2(N)
      REAL F,G,H,SCALE
      DO 100 I = 1, N
  100 D(I) = A(I,I)
C     ********** FOR I=N STEP -1 UNTIL 1 DO -- **********
      DO  300 II = 1, N
         I = N + 1 - II
         L = I - 1
         H = 0.0
         SCALE = 0.0
         IF (L .LT. 1) GO TO 130
C     ********** SCALE ROW (ALGOL TOL THEN NOT NEEDED) **********
         DO 120 K = 1, L
  120    SCALE = SCALE + ABS(A(I,K))
C
         IF (SCALE .NE. 0.0) GO TO 140
  130    E(I) = 0.0
         E2(I) = 0.0
         GO TO 290
C
  140    DO 150 K = 1, L
            A(I,K) = A(I,K) / SCALE
            H = H + A(I,K) * A(I,K)
  150    CONTINUE
C
         E2(I) = SCALE * SCALE * H
         F = A(I,L)
         G = -SIGN(SQRT(H),F)
         E(I) = SCALE * G
         H = H - F * G
         A(I,L) = F - G
         IF (L .EQ. 1) GO TO 270
         F = 0.0
C
         DO 240 J = 1, L
            G = 0.0
C     ********** FORM ELEMENT OF A*U **********
            DO 180 K = 1, J
  180       G = G + A(J,K) * A(I,K)
C
            JP1 = J + 1
            IF (L .LT. JP1) GO TO 220
C
            DO 200 K = JP1, L
  200       G = G + A(K,J) * A(I,K)
C     ********** FORM ELEMENT OF P **********
  220       E(J) = G / H
            F = F + E(J) * A(I,J)
  240    CONTINUE
C
         H = F / (H + H)
C     ********** FORM REDUCED A **********
         DO 260 J = 1, L
            F = A(I,J)
```

```
            G = E(J) + H * F
            E(J) = G
C
            DO 260 K = 1, J
               A(J,K) = A(J,K) + F * E(K) + G * A(I,K)
  260    CONTINUE
C
  270    DO 280 K = 1, L
  280    A(I,K) = SCALE * A(I,K)
C
  290    H = D(I)
         D(I) = A(I,I)
         A(I,I) = H
  300 CONTINUE
C
      RETURN
C     ********** LAST CARD OF TRED1 **********
      END
```

```
PROGRAM TREDEEN(INPUT,OUTPUT);
 (*ST-,P-*)
    CONST  N=25;
 TYPE DIM = ARRAY[1..N] OF REAL;
      DIMDIM = ARRAY[1..N,1..N] OF REAL;
VAR I,J,K,NN:INTEGER;=
 TIM:REAL;
    ITE ,=MAX : INTEGER;
     TOL:REAL;
 A , COPYA : DIMDIM;
 D,E,E2 : DIM;
PROCEDURE TRED1(VAR N:INTEGER;TOL:REAL;VAR A:DIMDIM;VAR D,E,E2:DIM);
  (* TRANSLATION OF TRED1 (WILKINSON AND REINSCH)*)

LABEL 100;
 VAR I,J,K,L:INTEGER;
     F,G,H:REAL;
 BEGIN
  FOR I:=1 TO N DO D[I]:=A[I,I];
  FOR I:=N DOWNTO 1 DO
   BEGIN L:=I-1; H:=0;
   =       FOR K:=1 TO L DO H:=H+A[I,K]*A[I,K];
  IF H <= TOL THEN
   BEGIN  E[I]:=0; E2[I]:=0;
          GOTO 100
   END;
  E2[I]:=H; F:=A[I,I-1];
  IF F >= 0 THEN E[I]:=-SQRT(H) ELSE E[I]:=SQRT(H);
  G:=E[I];
 H:=H-F*G;
 A[I,I-1]:=F-G; F:=0;
 FOR J:=1 TO L DO
  BEGIN G:=0;
   FOR K:=1 TO J DO G:=G+A[J,K]*A[I,K];=
   FOR K:=J+1 TO L DO
      G:=G+A[K,J]*A[I,K];
    G:=G/H;
    E[J]:=G;
    F:=F+G*A[I,J];
   END (* FOR J*);
  H:=F/(H+H);
  FOR J:=1 TO L DO
   BEGIN F:=A[I,J];
     G:=E[J]-H*F;
    E[J]:=G;
   FOR K:=1 TO J DO
   A[J,K]:=A[J,K]-F*E[K]-G*A[I,K];
   END (* FOR J*);

 100 : H:=D[I];
    D[I]:=A[I,I];
   A[I,I]:=H;
  END (* FOR I*);
END (* TRED1*);

   BEGIN

WRITE('1 DIMENSIE MATRIX',N);
WRITELN;
 FOR I:=1 TO N DO
 FOR J:=1 TO N DO
IF I>J THEN COPYA[I,J]:=I ELSE COPYA[I,J]:=J;
 TOL:=1.0E-200;

NN:=N;
MAX:=10;

TIM:=CLOCK;
FOR ITE:=1 TO MAX DO
 BEGIN
FOR I:=1 TO N DO
FOR J:=1 TO N DO A[I,J]:=COPYA[I,J];
TRED1(NN,TOL,A,D,E,E2);
 END;
TIM:=CLOCK-TIM;

WRITE('ODE DIAGONAAL IS:');
 WRITELN(' ');
 FOR I:=1 TO N DO WRITE(D[I]:14:7);
 WRITELN;
```

```
 WRITELN(' EN DE NEVENDIAGONAAL IS:');
 WRITELN('       ');
 FOR I :=1 TO N DO WRITE(E[I]:14:6);
WRITELN;
WRITE  ('OCP-TME:  ');WRITE(TIM:10:4);
END.
```

```
"BEGIN""COMMENT" JKOK, 750402, LSQPROCS TIJD=TEST ;

  "INTEGER" I, J, N, M;
  "ARRAY" AUX[2 : 5];
  "REAL" H, KLOK;

  "REAL""PROCEDURE" VECVEC(L, U, S, A, B); "CODE" 34010;
  "REAL""PROCEDURE" MATVEC(L, U, I, A, B); "CODE" 34011;
  "PROCEDURE" LSQORTDEC(A, N, M, AUX, AID, CI); "CODE" 34134;
  "PROCEDURE" LSQSOL(A, N, M, AID, CI, B); "CODE" 34131;
  "PROCEDURE" LSQDGLINV(A, M, AID, CI, DIAG); "CODE" 34132;

  "COMMENT" START PROGRAM ; AUX[2]:= 1,0"=14;
  KLOK:= CLOCK;
  "FOR" N:= 10 "STEP" 10 "UNTIL" 60 "DO"
  "FOR" M:= 1 "STEP" 1 "UNTIL" 15 "DO"
  "IF" N >= M "THEN"
  "BEGIN" "ARRAY" A, C[1 : N, 1 : M], B, X[1 : N], DIAG, AID[1 : M];
      "INTEGER""ARRAY" PIV[1 : M];

      "FOR" J:= 1 "STEP" 1 "UNTIL" M "DO"
      "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
      "BEGIN" H:= 1 / (I + J - 1); A[I,J]:= H; C[I,J]:= H "END";
      "FOR" J:= 1 "STEP" 1 "UNTIL" M "DO" X[J]:= J;
      "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
         B[I]:= MATVEC(1, M, I, A, X);
      "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO" X[I]:= B[I];
      OUTPUT(61, "("2(6S, 3ZD)")", "("  N = ")", N, "(" , M = ")", M);

      LSQORTDEC(A, N, M, AUX, AID, PIV);
      "IF" AUX[3] = M "THEN"
      "BEGIN" LSQSOL(A, N, M, AID, PIV, X);
         LSQDGLINV(A, M, AID, PIV, DIAG);
         OUTPUT(61, "("""(" AUX[EPS,RNK,MAXI] = ")", B=D,3D"+3D,
           5ZD, B=D,3D"+3D, /")", AUX[2], AUX[3], AUX[5]);
         "FOR" J:= 1 "STEP" 1 "UNTIL" M "DO"
         "BEGIN" OUTPUT(61, "("2B, B=3ZD,4D")", X[J]);
            "IF" J = 8 "OR" J = M "THEN" OUTPUT(61, "("/")")
         "END";
         OUTPUT(61, "("21S, B=D,3D"+3D, /")",
            "("RESIDUE (DELIVERED) :")", SQRT(VECVEC(M + 1, N, 0,
            X, X)));
         H:= 0; "FOR" I:= 1 "STEP" 1 "UNTIL" N "DO"
         H:= (B[I] - MATVEC(1, M, I, C, X)) ** 2 + H;
         OUTPUT(61, "("21S, B=D,3D"+3D")",
            "("RESIDUE (CHECKED)   :")", SQRT(H));
      "END" "ELSE" OUTPUT(61, "(" 4B"("RANK = ")", 3ZD")", AUX[3]);
      OUTPUT(61, "("21B"("KLOK :")", =B5ZD,3D,"(" SEC,")",/")",
         CLOCK - KLOK)
  "END"
"END"
      "EOP"
```

```
NUMPRC ;
'BEGIN' # JKOK, 750317, CREATE LEAST SQUARES PROCEDURES PRELUDE #

  'MODE' 'TOLS' = 'STRUCT' ('REAL' PREC, MAX),

  'OP' * = ('REF' []'REAL' A, B) 'REAL' :
    ('REAL' S:= 0; 'FOR' I 'TO' 'UPB' A 'DO' S +:= A[I] * B[I]'OD'; S),

  'OP' * = ('REAL' A, 'REF' []'REAL' B) []'REAL' :
    ([1 : 'UPB' B]'REAL' C;
      'FOR' I 'TO' 'UPB' B 'DO' C[I]:= A * B[I]'OD'; C
    ),

  'OP' +:= = ('REF'[]'REAL' A, []'REAL' B) 'REF'[]'REAL' :
    ('FOR' I 'TO' 'UPB' A 'DO' A[I] +:= B[I] 'OD'; A);

  'PROC' LSQDEC = ('REF'[,]'REAL' A, 'REF''TOLS' AUX,
     'REF'[]'REAL' AID, 'REF'[]'INT' CI) 'INT' :
  'IF''INT' N = 1 'UPB' A, M = 2 'UPB' A;
     'UPB' AID /= M 'OR' 'UPB' CI /= M 'THEN' = 1
  'ELSE''INT' R:= 0, MIMN:= (M < N | M | N), PK:= 1,
      'REAL' W, EPS, SIGMA:= 0, AIDK, BETA, [1 : M]'REAL' SUM;

      'FOR' K 'TO' M
      'DO''IF' (W:= SUM[K]:= A[ ,K] * A[ ,K]) > SIGMA
         'THEN' SIGMA:= W; PK:= K 'FI'
      'OD';

      W:= MAX 'OF' AUX:= SQRT(SIGMA); EPS:= (PREC 'OF' AUX) * W;
      'FOR' K 'TO' MINMN 'WHILE' W >= EPS
      'DO''REAL' AKK= A[K,PK]; R:= K; CI[K]:= PK;
         'IF' PK /= K
         'THEN' []'REAL' H= A[ ,K]; A[ ,K]:= A[ ,PK];
            A[ ,PK]:= H; SUM[PK]:= SUM[K]
         'FI';
         AIDK:= AID[K]:= (AKK < 0 | W | = W); A[K,K]:= AKK = AIDK;
         BETA:= = 1 / (SIGMA = AKK * AIDK); PK:= K; SIGMA:= 0;
        'FOR' J 'FROM' K + 1 'TO' M
        'DO' A[K : ,J] +:= BETA * (A[K: ,K] * A[K: ,J]) * A[K: ,K];
           'IF' (W:= SUM[J] =:= A[K,J] ** 2) > SIGMA
           'THEN' PK:= J; SIGMA:= W 'FI'
        'OD';
        W:= SQRT( SIGMA:= A[K + 1 : , PK] * A[K + 1 : , PK] )
      'OD';
      R
  'FI' # END OF HOUSEHOLDER TRIANGULARIZATION # ,

  'PROC' LSQSOL = ('REF' [,]'REAL' A, 'REF' []'REAL' AID, 'REF' [] 'INT'
        CI, 'REF' []'REAL' B) []'REAL';
  'BEGIN' 'INT' N = 1 'UPB' A, M = 2 'UPB' A, 'INT' CIK;
    [1:N]'REAL' BB:= B;

    'IF' M <= N
    'THEN''FOR' K 'TO' M 'DO' BB[K: ] +:=
       A[K: ,K] * BB[K: ] / (AID[K] * A[K,K]) * A[K: ,K] 'OD';
       'FOR' K 'FROM' M 'BY' = 1 'TO' 1 'DO' BB[K] :=
          (BB[K] = A[K,K+1: ] * BB[K+1:M]) / AID[K] 'OD';
       'FOR' K 'FROM' M = 1 'BY' = 1 'TO' 1
       'DO''IF' CIK:= CI[K]; CIK /= K
          'THEN''REAL' W= BB[K]; BB[K]:= BB[CIK]; BB[CIK]:= W 'FI'
       'OD'
    'FI';
    BB
  'END' # OF COMPUTATION OF LEAST SQUARES SOLUTION #,

  'PROC' LSQDGLINV = ('REF' [,]'REAL' A, 'REF' [] 'REAL' AID,
        DIAG, 'REF' [] 'INT' CI) 'VOID' :
  'BEGIN''INT' M = 2 'UPB' A; 'FOR' K 'TO' M
     'DO' DIAG[K]:= 1 / AID[K];
        'FOR' J 'FROM' K + 1 'TO' M
        'DO' DIAG[J]:= = (A[K : J = 1, J] * DIAG[K : J = 1]) / AID[J]
        'OD';
        DIAG[K]:= DIAG[K : ] * DIAG[K : ]
     'OD';
     'FOR' K 'FROM' M 'BY' = 1 'TO' 1
     'DO' 'INT' CIK:= CI[K]; 'IF' CIK /= K
        'THEN' 'REAL' W:= DIAG[K]; DIAG[K]:= DIAG[CIK]; DIAG[CIK]:= W
        'FI'
     'OD'
  'END' # LSQDGLINV #,
```

```
  'PROC' LSQDECSOL = ('REF'[,] 'REAL' A, 'REF''TOLS' AUX,
          'REF' [] 'REAL' DIAG, B) []'REAL' ;
  'IF' 'INT' M = 2 'UPB' A; [1 : M]'REAL' AID, [1 :-M]'INT' CI;
       LSQDEC(A, AUX, AID, CI) = M
  'THEN' LSQDGLINV(A[ : M, ], AID, DIAG, CI);
       LSQSOL(A, AID, CI, B)
  'FI' # LSQDECSOL #;

  'PR' PROG 'PR' 'SKIP'
'END' #OF PRELUDE AND POSTLUDE NUMERICAL PROCEDURES# 

'BEGIN' # JKOK, 750317, TEST PROGRAM LEAST SQUARES PROCEDURES IN
           LIBRARY PRELUDE #
    'TOLS' AUX; PREC 'OF' AUX := 1.0E-14;
    'REAL' KLOK:= CLOCK;
    'FOR' N 'FROM' 10 'BY' 10 'TO' 60
    'DO''FOR' M 'TO' 15 'WHILE' M <= N
      'DO' [1 : N, 1 : M]'REAL' A, C, [1 : N]'REAL' B, X;
        'FOR' J 'TO' M 'DO''FOR' I 'TO' N
        'DO' A[I,J]:= (C[I,J]:= 1 / (I + J - 1)) 'OD''OD';
        'FOR' J 'TO' M 'DO' X[J]:= J 'OD';
        'FOR' I 'TO' N
        'DO' B[I]:= A[I, : M] * X[ : M] 'OD';
        X[1 : N]:= B[1 : N];
        PRINT(("  N = ", WHOLE(N, 4), " , M = ", WHOLE(M, 4)));

        'IF' [1 : M]'REAL' AID, [1 : M]'INT' PIV, 'INT' RNK;
           ( RNK:= LSQDEC(A, AUX, AID, PIV) ) = M
        'THEN' [1 : N] 'REAL' SOL:= LSQSOL(A, AID, PIV, X);
           'REAL' H := 0, [1 : M]'REAL' DIAG;
           LSQDGLINV(A[ : M, ], AID, DIAG, PIV);
           PRINT((" AUX[EPS,RNK,MAXI] =  ", FLOAT(PREC 'OF' AUX, 12, 3,
              4), WHOLE(M, 4), FLOAT(MAX 'OF' AUX, 11, 3, 4), NEWLINE));
           'FOR' J 'TO' M
           'DO' PRINT( FIXED(SOL[J], 15, 4) );
              'IF' J = 8 'OR' J = M 'THEN' PRINT(NEWLINE) 'FI'
           'OD';
           PRINT(("RESIDUE (DELIVERED) : ", FLOAT(SQRT(SOL[M + 1 : N] *
              SOL[M + 1 : N]), 11, 3, 4), NEWLINE));
           'FOR' I 'TO' N
           'DO' H+:= (B[I] - C[I, ] * SOL[ : M]) ** 2 'OD';
           PRINT(("RESIDUE (CHECKED)   : ", FLOAT(SQRT(H), 11, 3, 4)))
        'ELSE' PRINT(("    RANK = ", WHOLE(RNK, 4))) 'FI';
        PRINT((" KLOK : ", FIXED(CLOCK - KLOK, 15, 4),
           " SEC.", NEWLINE))
      'OD'
    'OD'
 'END'
```

```
C$    TRACE SUBSCRIPTS
C
      PROGRAM MAIN (OUTPUT, TAPE1 = OUTPUT, DEBUG = OUTPUT)
C
C*****************************************************
C                                                     *
C       J   K O K    ,   M  C                       *
C                                                     *
C       BEPALING REKENTIJDEN LSQ-PROBLEMEN           *
C                                                     *
C       750401                                        *
C                                                     *
C*****************************************************
C
      DIMENSION A(60,15), C(60,15), X(60), B(60), DIAG(15), AID(15),
     1   SA(15), IPR(15), AUX(5)
      REAL KLOK
C
C     INITIALISEER KLOK
C
      CALL SECOND(KLOK)
      AUX(2) = 1.E-14
C
      DO 1 N = 10, 60, 10
         DO 2 M = 1, 15
            IF (N - M) 2, 3, 3
C
    3       DO 4 J = 1, M
            DO 4 I = 1, N
               H = 1. / FLOAT(I + J - 1)
               A(I,J) = H
    4          C(I,J) = H
C
            DO 5 J = 1, M
    5          X(J) = J
C
            DO 6 I = 1, N
               H = 0.
               DO 7 J = 1, M
    7             H = A(I,J) * X(J) + H
    6          B(I) = H
C
            DO 8 I = 1, N
    8          X(I) = B(I)
C
  101       FORMAT(6H  N = ,I4, 7H , M = ,I4, E12.5, I4, E12.5)
C
            CALL LSQDEC(A, 60, N, M, AUX, AID, SA, IPR)
            WRITE(1, 101) N, M, AUX(2), INT(AUX(3)), AUX(5)
            IF (INT(AUX(3)) - M) 99, 9, 99
C
    9       CALL LSQSOL(A, 60, N, M, AID, IPR, X)
            CALL LSQDGL(A, 60, M, AID, IPR, DIAG)
C
  102       FORMAT(4H X =, (1X,8F10.4))
            WRITE(1, 102) (X(J), J = 1, M)
C
            H = 0.
            M1 = M + 1
            IF (N - M1) 14, 15, 15
   15       DO 11 I = M1, N
   11         H = H + X(I) ** 2
            H = SQRT(H)
  103       FORMAT(24H RESIDUE (DELIVERED) :  ,E12.5)
  104       FORMAT(24H RESIDUE (CHECKED)   :  ,E12.5, 10H  KLOK :  ,
     1         F10.3, 5H SEC.)
  105       FORMAT(20H SINGULIER, KLOK :  ,F10.3, 5H SEC.)
C
   14       WRITE(1, 103) H
            H = 0.
            DO 12 I = 1, N
               H1 = 0.
               DO 13 J = 1, M
   13             H1 = H1 + C(I,J) * X(J)
   12          H = H + (B(I) - H1) ** 2
C
            H = SQRT(H)
            CALL SECOND(H1)
            H1 = H1 - KLOK
            WRITE(1, 104) H, H1
```

```
            GO TO 2
C
   99       CALL SECOND(H1)
            H1 = H1 - KLOK
            WRITE(1, 105) H1
C
    2       CONTINUE
    1    CONTINUE
      STOP
      END
      SUBROUTINE LSQDGL
     1   (A, NR, MM, AID, IPR, DIAG)
C$    TRACE SUBSCRIPTS
      DIMENSION A(NR, MM), AID(MM), IPR(MM), DIAG(MM)
C
C     LSQDGLINV, SEE NUMAL
C
      M = MM
      DO 49 K = 1, M
         DIAG(K) = 1. / AID(K)
         K1 = K + 1
         IF (M - K1) 47, 43, 43
   43    DO 46 J = K1, M
            H = 0.
            J1 = J - 1
            IF (J1 - K) 46, 44, 44
   44       DO 45 I = K, J1
   45          H = A(I,J) * DIAG(I) + H
   46       DIAG(J) = - H / AID(J)
   47    H = 0.
         DO 48 I = K, M
   48       H = H + DIAG(I) ** 2
   49    DIAG(K) = H
C
      K = M + 1
      DO 51 L = 1, M
         K = K - 1
         IPIV = IPR(K)
         IF (K - IPIV) 50, 51, 51
   50    H = DIAG(K)
         DIAG(K) = DIAG(IPIV)
         DIAG(IPIV) = H
   51    CONTINUE
      RETURN
      END
      SUBROUTINE  LSQDEC
     1   (A, NR, NN, MM, AUX, AID, SUM, CI)
C$    TRACE SUBSCRIPTS
C
C     LSQORTDEC, SEE  NUMAL
C
      DIMENSION A(NR, MM), AUX(5), AID(MM), SUM(MM)
      INTEGER CI(MM)
      REAL NORM
C
      N = NN
      M = MM
      MINMN = M
      IF (N .LT. M) MINMN = N
      AUX(3) = MINMN
C
      NORM = 0.
      DO 2 K = 1, M
         H = 0.
         DO 1 I = 1, N
    1       H = A(I,K) ** 2 + H
         SUM(K) = H
    2    IF (H .GT. NORM) NORM = H
C
      W = SQRT(NORM)
      AUX(5) = W
      EPS = AUX(2) * W
C
C
      DO 20 K = 1, MINMN
         SIGMA = SUM(K)
         KPIV = K
C
         IF (M - K) 8, 8, 3
    3    K1 = K + 1
```

```
      DO 4 J = K1, M
         IF (SUM(J) - SIGMA) 4, 4, 5
    5    SIGMA = SUM(J)
         KPIV = J
    4    CONTINUE
C
      IF (K - KPIV) 7, 8, 8
    7 SUM(KPIV) = SUM(K)
      DO 9 I = 1, N
         H = A(I,K)
         A(I,K) = A(I,KPIV)
    9    A(I,KPIV) = H
C
    8 CI(K) = KPIV
      AKK = A(K,K)
      SIGMA = 0.
      DO 11 I = K, N
   11    SIGMA = A(I,K) ** 2 + SIGMA
      W = SQRT(SIGMA)
C
      AIDK = W
      IF (AKK .GE. 0.) AIDK = - W
      AID(K) = AIDK
C
      IF (W .GE. EPS) GO TO 13
      AUX(3) = K - 1
      RETURN
C
   13 BETA = 1. / (SIGMA - AKK * AIDK)
      A(K,K) = AKK - AIDK
C
      IF (M - K) 20, 20, 14
   14 DO 19 J = K1, M
C
         H = 0.
         DO 15 I = K, N
   15       H = A(I,K) * A(I,J) + H
C
         H = - BETA * H
         DO 16 I = K, N
   16       A(I,J) = A(I,J) + H * A(I,K)
C
   19    SUM(J) = SUM(J) - A(K,J) ** 2
C
   20 CONTINUE
C
   RETURN
   END
   SUBROUTINE  LSQSOL
  1   (A, NR, N, MM, AID, CI, B)
C$ TRACE SUBSCRIPTS
C
C    LSQSOL, SEE NUMAL
C
      DIMENSION A(NR, MM), AID(MM), B(N)
      INTEGER CI(MM)
C
      M = MM
      DO 3 K = 1, M
         H = 0.
         DO 2 I = K, N
    2       H = A(I,K) * B(I) + H
         H = H / (AID(K) * A(K,K))
         DO 1 I = K, N
    1       B(I) = H * A(I,K) + B(I)
    3 CONTINUE
C
      K = M + 1
      DO 6 L = 1, M
         K1 = K
         K = K - 1
         IF (M - K) 6, 6, 4
    4    H = 0.
         DO 5 J = K1, M
    5       H = A(K,J) * B(J) + H
         B(K) = B(K) - H
    6    B(K) = B(K) / AID(K)
C
      K = M + 1
      DO 8 L = 1, M
```

```
      K = K + 1
      IK = CI(K)
      IF (IK - K) 8, 8, 7
    7 H = B(K)
      B(K) = B(IK)
      B(IK) = H
    8 CONTINUE
C
      RETURN
      END
```

```
PROGRAM ELMPROCS (OUTPUT); (* VOOR TE VERTALEN PROCEDURES *)
(*$T+,E+ *)

    CONST ROWN = 60; COLM = 15;

    TYPE NMMAT = ARRAY [1 : ROWN, 1 : COLM] OF REAL;
       NVEC = ARRAY [1 : ROWN] OF REAL;
       MVEC = ARRAY [1 : COLM] OF REAL;

    FUNCTION TAMMAT(L, U, I, J : INTEGER; VAR A, B : NMMAT) : REAL;
       VAR S : REAL;
    BEGIN S:= 0;
       FOR L:= L TO U DO S:= A[L, I] * B[L, J] + S;
       TAMMAT:= S
    END (* TAMMAT *);

    PROCEDURE ELMCOL(L, U, I, J : INTEGER; VAR A, B : NMMAT;
          X : REAL);
    BEGIN FOR L:= L TO U DO A[L, I]:= A[L, I] + B[L, J] * X END;

    PROCEDURE ICHCOL(L, U, I, J : INTEGER; VAR A : NMMAT);
       VAR W : REAL;
    BEGIN FOR L:= L TO U DO
       BEGIN W:= A[L, I]; A[L, I]:= A[L, J]; A[L, J]:= W END
    END (* ICHCOL *);

    FUNCTION MATVEC(L, U, I : INTEGER; VAR A : NMMAT; VAR B : NVEC)
          : REAL;
       VAR S : REAL;
    BEGIN S:= 0;
       FOR L:= L TO U DO S:= A[I, L] * B[L] + S;
       MATVEC:= S
    END (* MATVEC *);

    FUNCTION TAMVEC(L, U, I : INTEGER; VAR A : NMMAT; VAR B : NVEC)
          : REAL;
       VAR S : REAL;
    BEGIN S:= 0;
       FOR L:= L TO U DO S:= A[L, I] * B[L] + S;
       TAMVEC:= S
    END (* TAMVEC *);

    PROCEDURE ELMVECCOL(L, U, I : INTEGER; VAR A : NVEC;
          VAR B : NMMAT; X : REAL);
    BEGIN FOR L:= L TO U DO A[L]:= B[L, I] * X + A[L] END;

    FUNCTION VECVEC(L, U, S : INTEGER; VAR A, B : NVEC) : REAL;
       VAR X : REAL;
    BEGIN X:= 0;
       FOR L:= L TO U DO X:= A[L] * B[L + S] + X;
       VECVEC:= X
    END (* VECVEC *);

BEGIN (* DUMMY BODY *) END.

PROGRAM NUMPROCS (INPUT, OUTPUT); (* VOOR TE VERTALEN PROCEDURES *)
(*$T+,E+,    LSQORTDEC, 750113 *)
       CONST ROWN = 60; COLM = 15;

       TYPE NMMAT = ARRAY [1 : ROWN, 1 : COLM] OF REAL;
          NVEC = ARRAY [1 : ROWN] OF REAL;
          MVEC = ARRAY [1 : COLM] OF REAL;
          MINTVEC = ARRAY [1 : COLM] OF INTEGER;
          RECAUX = RECORD EPS, MAXAII : REAL; RNK : INTEGER END;

    PROCEDURE LSQORTDEC(VAR A : NMMAT; N, M : INTEGER;
          VAR AUX : RECAUX; VAR AID : MVEC; VAR CI : MINTVEC);

       LABEL 1;
       VAR J, K, KPIV : INTEGER;
          BETA, SIGMA, NORM, W, EPS, AKK, AIDK : REAL;
          SUM : ARRAY [1 : COLM] OF REAL;

       FUNCTION TAMMAT(L, U, I, J : INTEGER; VAR A, B : NMMAT) : REAL;
       EXTERN;
       PROCEDURE ELMCOL(L, U, I, J : INTEGER; VAR A, B : NMMAT;
            X : REAL); EXTERN;
       PROCEDURE ICHCOL(L, U, I, J : INTEGER; VAR A : NMMAT); EXTERN;

    BEGIN NORM:= 0; AUX.RNK:= M;
```

BIBLIOTHEEK MATHEMATISCH CENTRUM
AMSTERDAM

```
      FOR K:= 1 TO M DO
      BEGIN W:= TAMMAT(1, N, K, K, A, A); SUM[K]:= W;
          IF W > NORM THEN NORM:= W
      END;
      W:= SQRT(NORM); AUX.MAXAII:= W; EPS:= AUX.EPS * W;
      FOR K:= 1 TO M DO
      BEGIN SIGMA:= SUM[K]; KPIV:= K;
          FOR J:= K + 1 TO M DO
          IF SUM[J] > SIGMA THEN
          BEGIN SIGMA:= SUM[J]; KPIV:= J END;
          IF KPIV <> K THEN
          BEGIN SUM[KPIV]:= SUM[K]; ICHCOL(1, N, K, KPIV, A) END;
          CI[K]:= KPIV; AKK:= A[K,K];
          SIGMA:= TAMMAT(K, N, K, K, A, A); W:= SQRT(SIGMA);
          IF AKK < 0 THEN AIDK:= W ELSE AIDK:= - W; AID[K]:= AIDK;
          IF W < EPS THEN
          BEGIN AUX.RNK:= K - 1; GOTO 1 END;
          BETA:= 1 / (SIGMA - AKK * AIDK); A[K,K]:= AKK - AIDK;
          FOR J:= K + 1 TO M DO
          BEGIN ELMCOL(K, N, J, K, A, A, - BETA * TAMMAT(K, N,
              K, J, A, A)); SUM[J]:= SUM[J] - SQR(A[K,J])
          END
      END (* FOR K *);
  1 :
  END (* LSQORTDEC *);

  PROCEDURE LSQDGLINV(VAR A : NMMAT; M : INTEGER; VAR AID, DIAG
        : MVEC; VAR CI : MINTVEC);

     VAR J, K, CIK : INTEGER;
        W : REAL;

     FUNCTION VECVEC(L, U, S : INTEGER; VAR A, B : MVEC) : REAL;
          EXTERN;
     FUNCTION TAMVEC(L, U, I : INTEGER; VAR A : NMMAT; VAR B : MVEC)
        : REAL; EXTERN;

  BEGIN FOR K:= 1 TO M DO
      BEGIN DIAG[K]:= 1 / AID[K];
          FOR J:= K + 1 TO M DO
          DIAG[J]:= - TAMVEC(K, J - 1, J, A, DIAG) / AID[J];
          DIAG[K]:= VECVEC(K, M, 0, DIAG, DIAG)
      END;
      FOR K:= M DOWNTO 1 DO
      BEGIN CIK:= CI[K]; IF CIK <> K THEN
          BEGIN W:= DIAG[K]; DIAG[K]:= DIAG[CIK]; DIAG[CIK]:= W END
      END
  END (* LSQDGLINV *);

  PROCEDURE LSQSOL(VAR A : NMMAT; N, M : INTEGER; VAR AID : MVEC;
        VAR CI : MINTVEC; VAR B : NVEC);

     VAR K, CIK : INTEGER;
        W : REAL;

     FUNCTION MATVEC(L, U, I : INTEGER; VAR A : NMMAT; VAR B : NVEC)
        : REAL; EXTERN;
     FUNCTION TAMVEC(L, U, I : INTEGER; VAR A : NMMAT; VAR B : NVEC)
        : REAL; EXTERN;
     PROCEDURE ELMVECCOL(L, U, I : INTEGER; VAR A : NVEC;
          VAR B : NMMAT; X : REAL); EXTERN;

  BEGIN FOR K:= 1 TO M DO
     BEGIN W:= TAMVEC(K, N, K, A, B) / (AID[K] * A[K,K]);
        ELMVECCOL(K, N, K, B, A, W)
     END;
     FOR K:= M DOWNTO 1 DO B[K]:= (B[K] - MATVEC(K + 1, M, K, A, B))
        / AID[K];
     FOR K:= M DOWNTO 1 DO
     BEGIN CIK:= CI[K]; IF CIK <> K THEN
        BEGIN W:= B[K]; B[K]:= B[CIK]; B[CIK]:= W END
     END
  END (* LSQSOL *);

  PROCEDURE LSQDECSOL(VAR A : NMMAT; N, M : INTEGER;
        VAR AUX : RECAUX; VAR DIAG : MVEC; VAR B : NVEC);
     VAR AID : ARRAY [1:COLM] OF REAL; (* COLM IS CONSTANT *)
        CI : ARRAY [1 : COLM] OF INTEGER;

     PROCEDURE LSQORTDEC(VAR A : NMMAT; N, M : INTEGER;
```

```
            VAR AUX : RECAUX; VAR AID : MVEC; VAR CI : MINTVEC);
         EXTERN;
      PROCEDURE LSQDGLINV(VAR A : NMMAT; M : INTEGER;
            VAR AID, DIAG : MVEC; VAR CI : MINTVEC); EXTERN;
      PROCEDURE LSQSOL(VAR A : NMMAT; N, M : INTEGER; VAR AID : MVEC;
            VAR CI : MINTVEC; VAR B : NVEC); EXTERN;

   BEGIN LSQORTDEC(A, N, M, AUX, AID, CI);
      IF AUX.RNK = M THEN
      BEGIN LSQDGLINV(A, M, AID, DIAG, CI);
         LSQSOL(A, N, M, AID, CI, B)
      END
   END (* LSQDECSOL *);

BEGIN (* DUMMY BODY *) END .

(*ST+ , JKOK, 750313, LSQPROCS TIJD=TEST *)
   PROGRAM EXAMPLE (INPUT, OUTPUT);
      (* JKOK, 730912, TEST LSQORTDEC, LSQSOL, LSQDGLINV *)

      CONST NN = 60; MM = 15;

      TYPE NMMAT = ARRAY [1 : NN, 1 : MM] OF REAL;
         NVEC = ARRAY [1 : NN] OF REAL;
         MVEC = ARRAY [1 : MM] OF REAL;
         MINTVEC = ARRAY [1 : MM] OF INTEGER;
         RECAUX = RECORD EPS, MAXAII : REAL; RNK : INTEGER END;

      VAR I, J, KLOK, N, M : INTEGER;
         A, C : NMMAT;
         B, X : NVEC;
         DIAG, AID : MVEC;
         AUX : RECAUX;
         PIV : MINTVEC;
         H : REAL;

      FUNCTION VECVEC(L, U, S : INTEGER; VAR A, B : NVEC) : REAL;
            EXTERN;
      FUNCTION MATVEC(L, U, I : INTEGER; VAR A : NMMAT;
         VAR B : NVEC) : REAL; EXTERN;
      PROCEDURE LSQORTDEC(VAR A : NMMAT; N, M : INTEGER; VAR AUX
            : RECAUX; VAR AID : MVEC; VAR CI : MINTVEC); EXTERN;
      PROCEDURE LSQSOL(VAR A : NMMAT; N, M : INTEGER; VAR AID : MVEC;
            VAR CI : MINTVEC; VAR B : NVEC); EXTERN;
      PROCEDURE LSQDGLINV(VAR A : NMMAT; M : INTEGER; VAR AID, DIAG :
         MVEC; VAR CI : MINTVEC); EXTERN;

BEGIN (* START PROGRAM *) AUX.EPS:= 1.0E-14;
   N:= 0; KLOK:= CLOCK;
   REPEAT N:= N + 10; FOR M:= 1 TO MM DO
     IF N >= M THEN
     BEGIN
       FOR J:= 1 TO M DO FOR I:= 1 TO N DO
       BEGIN H:= 1 / (I + J - 1); A[I,J]:= H; C[I,J]:= H END;
       FOR J:= 1 TO M DO X[J]:= J;
       FOR I:= 1 TO N DO
       B[I]:= MATVEC(1, M, I, A, X);
       FOR I:= 1 TO N DO X[I]:= B[I];
       WRITE('  N = ', N : 4, ' , M = ', M : 4);
       LSQORTDEC(A, N, M, AUX, AID, PIV);
       IF AUX.RNK = M THEN
       BEGIN LSQSOL(A, N, M, AID, PIV, X);
          LSQDGLINV(A, M, AID, DIAG, PIV);
          WRITELN(' AUX(EPS,RNK,MAXI) = ', AUX.EPS : 12, AUX.RNK : 6,
            AUX.MAXAII : 12);
          FOR J:= 1 TO M DO
          BEGIN WRITE('  ', X[J] : 10 : 4);
             IF (J = 8) OR (J = M) THEN WRITELN
          END;
          WRITELN(' RESIDUE (DELIVERED) :', SQRT(VECVEC(M + 1, N, 0,
             X, X)) : 13);
          H:= 0; FOR I:= 1 TO N DO
          H:= SQR(B[I] - MATVEC(1, M, I, C, X)) + H;
          WRITE(' RESIDUE (CHECKED)   :', SQRT(H) : 13)
       END ELSE WRITE('     RANK = ', AUX.RNK : 4);
       WRITELN(' ' : 20, ' KLOK : ', (CLOCK - KLOK) / 1000 : 12 : 3,
          ' SEC.')
     END
   UNTIL N = NN
END .
```

```
"BEGIN" "INTEGER" K; "REAL" T, TOTALTIME; "ARRAY" A[1 : 2000];

    "PROCEDURE" MERGE(A, LOW, M, UP);
    "VALUE" LOW, M, UP; "INTEGER" LOW, M, UP; "ARRAY" A;
    "IF" LOW <= M "AND" M < UP "THEN"
    "BEGIN" "INTEGER" J, P, Q; "ARRAY" B[LOW : M];
        "FOR" J:= LOW "STEP" 1 "UNTIL" M "DO" B[J]:= A[J];
        P:= M + 1; Q:= LOW; J:= LOW - 1;
    NEXT:   J:= J + 1;
        "IF" B[Q] <= A[P] "THEN"
        "BEGIN" A[J]:= B[Q]; Q:= Q + 1; "IF" Q <= M "THEN" "GOTO" NEXT
        "END"
        "ELSE"
        "BEGIN" A[J]:= A[P]; P:= P + 1; "IF" P <= UP "THEN" "GOTO" NEXT;
            "FOR" J:= J + 1 "STEP" 1 "UNTIL" UP "DO"
            "BEGIN" A[J]:= B[Q]; Q:= Q + 1 "END"
        "END"
    "END" MERGE;

    "PROCEDURE" SORT(A, LOW, UP);
    "VALUE" LOW, UP; "INTEGER" LOW, UP; "ARRAY" A;
    "IF" LOW < UP "THEN"
    "BEGIN" "INTEGER" M;
       M:= (LOW + UP - 1) // 2;
       SORT(A, LOW, M); SORT(A, M + 1, UP);
       MERGE(A, LOW, M, UP)
   "END" SORT;

    TOTALTIME:= CLOCK;
    "FOR" K:= 1 "STEP" 1 "UNTIL" 2000 "DO"
    A[K]:= ENTIER(ABS(SIN(K)) * 25);
    "FOR" K:= 1 "STEP" 1 "UNTIL" 200 "DO"
    "BEGIN" OUTPUT(61,"("2BZD2B")", A[K]);
        "IF" K // 20 * 20 = K "THEN" OUTPUT(61,"("/")")
    "END";
    T:= CLOCK;
    SORT(A, 1, 2000);
    T:= CLOCK - T;
    OUTPUT(61,"("3/")");
    "FOR" K:= 1 "STEP" 1 "UNTIL" 200 "DO"
    "BEGIN" OUTPUT(61,"("2BZD2B")", A[K]);
        "IF" K // 20 * 20 = K "THEN" OUTPUT(61,"("/")")
    "END";
    TOTALTIME:= CLOCK - TOTALTIME;
    OUTPUT(61,"("3/,"("TOTAALTIJD PROGRAMMA: ")", N, /,
       "("TIJD ZUIVER GEBRUIKT VOOR SORTEREN: ")"")", TOTALTIME, T)
"END"
```

```
'BEGIN'
'PROC' MERGESORT = ('REF'[]'REAL' A)'VOID';
   'BEGIN'
      'PROC' MERGE = ('REF'[]'REAL' A, 'INT' M)'VOID';
         ('INT' N = 'UPB' A;
          'IF' 1 <= M 'AND' M < N
          'THEN' [1 : M]'REAL' B:= A[ : M];
                 'INT' P:= M + 1, Q:= 1, J:= 0;
          NEXT:  J+:= 1;
                 'IF' B[Q] <= A[P]
                 'THEN' A[J]:= B[Q]; Q+:= 1; (Q <= M | NEXT)
                 'ELSE' A[J]:= A[P]; P+:= 1; (P <= N | NEXT);
                        A[J + 1 : ]:= B[Q : ]
                 'FI'
          'FI' ) # END OF MERGE #;

      'PROC' SORT = ('REF'[]'REAL' A)'VOID';
         ('INT' N = 'UPB' A;
          'IF' 1 < N
          'THEN' 'INT' M = N 'OVER' 2;
                 SORT(A[ : M]); SORT(A[M + 1 : ]);
                 MERGE(A, M)
          'FI' ) # END OF SORT #;

      'REF'[]'REAL' AA= A[ : ];
      SORT(AA)
   'END' # END OF MERGESORT #;


[1:2000]'REAL' A;
'REAL' TOTALTIME:= CLOCK;
'FOR' K 'TO' 2000
'DO' A[K]:= 'ENTIER'('ABS' SIN(K) * 25) 'OD';
'FOR' K 'TO' 200
'DO' PRINT(WHOLE(A[K], 6));
     'IF' K 'MOD' 20 = 0 'THEN' PRINT(NEWLINE) 'FI'
'OD';
PRINT((NEWLINE, NEWLINE, NEWLINE));
'REAL' T:= CLOCK;
MERGESORT(A);
T:= CLOCK - T;
'FOR' K 'TO' 200
'DO' PRINT(WHOLE(A[K * 10], 6));
     'IF' K 'MOD' 20 = 0 'THEN' PRINT(NEWLINE) 'FI'
'OD';
TOTALTIME:= CLOCK - TOTALTIME;
PRINT((NEWLINE, NEWLINE, NEWLINE, "TOTAALTIJD PROGRAMMA: ", TOTALTIME,
       NEWLINE, "TIJD ZUIVER GEBRUIKT VOOR SORTEREN: ",T))
'END'
```

```
PROGRAM MERGESORT( OUTPUT ); (* TJD741030 *)

CONST N= 2000;
TYPE VECTOR= ARRAY [1..N] OF INTEGER;
VAR A: VECTOR;   K: INTEGER;
    T, TOTALTIME: REAL;

PROCEDURE MERGE(LOW, M, UP: INTEGER; VAR A: VECTOR);
VAR J, P, Q: INTEGER;  B: VECTOR;
BEGIN IF (LOW <= M) AND (M < UP) THEN
   BEGIN FOR J:= LOW TO M DO B[J]:= A[J];
      P:= M + 1; Q:= LOW; J:= LOW - 1;
      REPEAT J:= J + 1;
         IF B[Q] <= A[P] THEN
         BEGIN A[J]:= B[Q]; Q:= Q + 1 END
         ELSE
         BEGIN A[J]:= A[P]; P:= P + 1;
            IF P > UP THEN
            FOR J:= J + 1 TO UP DO
            BEGIN A[J]:= B[Q]; Q:= Q + 1 END
         END
      UNTIL Q > M
   END
END;

PROCEDURE SORT(LOW, UP: INTEGER; VAR A: VECTOR);
VAR M: INTEGER;
BEGIN IF LOW < UP THEN
   BEGIN M:= (LOW + UP - 1) DIV 2;
      SORT(LOW, M, A); SORT(M + 1, UP, A);
      MERGE(LOW, M, UP, A)
   END
END;

BEGIN TOTALTIME:= CLOCK;
   FOR K:= 1 TO N DO
   BEGIN A[K]:= TRUNC(ABS(SIN(K)) * 25) END;
   FOR K:= 1 TO 200 DO
   BEGIN WRITE(A[K] : 6);
      IF K DIV 20 * 20 = K THEN WRITELN
   END;
   T:= CLOCK;
   SORT(1, N, A);
   T:= CLOCK - T;
   WRITELN; WRITELN; WRITELN;
   FOR K:= 1 TO 200 DO
   BEGIN WRITE(A[K * 10] : 6);
      IF K DIV 20 * 20 = K THEN WRITELN
   END;
   TOTALTIME:= CLOCK - TOTALTIME;
   WRITELN;WRITELN;WRITELN;
   WRITELN(' TOTAALTIJD PROGRAMMA: ', TOTALTIME);
   WRITE(' TIJD ZUIVER GEBRUIKT VOOR SORTEREN: ', T)
END.
```

```
"BEGIN" "COMMENT" BEREKENING PRIEMFUNCTIE;
    "COMMENT" "CHECKON" UIT,LAST,P,V;
    "INTEGER" N,SORTN,I,I1,X,LIM,SQUARE,PRX,FX,MAXFX;
    "INTEGER" "ARRAY" UIT[1:20],LAST,P,V[1:25];
    "REAL" TIJD;
    "INTEGER" "PROCEDURE" F(P); "VALUE" P; "INTEGER" P;
    "BEGIN" "INTEGER" TL,A,B,U,Q,C; "BOOLEAN" JA;
        "COMMENT" "CHECKON" J;
            "BOOLEAN" "ARRAY" J[1:P÷1];
        "FOR" A:=2 "STEP" 1 "UNTIL" P÷1 "DO" J[A]:="FALSE";
        A:=0; U:=1;
        "FOR" A:=A+U "WHILE" U<=P÷2 "DO"
        "BEGIN" U:=U+2; B:=A÷P;
            "IF" B>0 "THEN" A:=B; J[A]:="TRUE"
        "END";
        Q:=P//2; U:=2;
        "FOR" A:=A "WHILE" J[U] "DO" U:=U+1;
        TL:="IF" J[P÷1] "THEN" 2×U÷1 "ELSE" U;
        U:="IF" (U>Q÷TL) "AND" (U<=Q) "THEN" Q "ELSE" U+TL;
       "FOR" A:=U÷1 "WHILE" U<=Q "DO"
        "BEGIN" JA:=J[U]; B:=U+1;
            "FOR" C:=1 "WHILE" J[A] "EQUIV" JA "DO" A:=A÷1;
            "FOR" C:=1 "WHILE" J[B] "EQUIV" JA "DO" B:=B+1;
            "IF" B÷A÷1>TL "THEN" TL:=B÷A÷1;
            U:="IF" (B>Q÷TL) "AND" (B<=Q) "THEN" Q "ELSE" B+TL
       "END";
       F:=TL
    "END";
    "PROCEDURE" UITVOER(N); "VALUE" N; "INTEGER" N;
    "IF" N=10 "THEN" OUTPUT(61,"("10(7ZD,3ZD),/")",UIT) "ELSE"
    "BEGIN" "INTEGER" I;
        "FOR" I:=2 "STEP" 2 "UNTIL" 2×N "DO"
        OUTPUT(61,"("7ZD,3ZD")",UIT[I÷1],UIT[I]);
        OUTPUT(61,"("/")")
    "END";

    "PROCEDURE" PRIEM(X); "INTEGER" X;
    "BEGIN" "INTEGER" K; "BOOLEAN" PRIM;
        PRIM:="FALSE";
        "FOR" K:=1 "WHILE" "NOT" PRIM "DO"
        "BEGIN" X:=X+2; "IF" SQUARE<=X "THEN"
            "BEGIN" V[LIM]:=SQUARE ;LIM:=LIM+1; SQUARE:=P[LIM]**2
            "END"; PRIM:="TRUE";
            "FOR" K:=K+1 "WHILE" PRIM "AND" (K<LIM) "DO"
            "BEGIN" "IF" V[K]<X "THEN" V[K]:=V[K]+P[K];
                PRIM:=(X "=" V[K])
            "END";
        "END"
    "END";

    TIJD:=CLOCK; OUTPUT(61,"(""("TIJD=")",3ZD,3D,/")",TIJD);
    SORTN:=25;N:=4999;
    "FOR" I:=1 "STEP" 1 "UNTIL" SORTN "DO" LAST[I]:=0;
    X:=1; LIM:=1; SQUARE:=4; I:=0; P[1]:=2; PRX:=1; MAXFX:=0; I1:=0;
    "FOR" I1:=I1 "WHILE" X<N "DO"
    "BEGIN" PRIEM(X); I:=I+2; UIT[I÷1]:=X; FX:=F(X); UIT[I]:=FX;
         PRX:=PRX+1; "IF" PRX<SORTN "THEN" P[PRX]:=X;
         "IF" X>LAST[FX] "THEN" LAST[FX]:=X;
         "IF" FX>MAXFX "THEN" MAXFX:=FX;
         "IF" I=20 "THEN" "BEGIN" UITVOER(10); I:=0 "END";
    "END"; UITVOER(I//2);
    OUTPUT(61,"(""("LIJST VAN PRIEMGETALLEN WAARVOOR F HET ")",/,
          "("WAARDE N AANNAM")",//")");
    "FOR" X:=1 "STEP" 1 "UNTIL" MAXFX "DO"
    OUTPUT(61,"("2ZD,9ZD,/")",X,LAST[X]);
    OUTPUT(61,"(""("VERBRUIKTE TIJD")",3ZD,3D,/")",CLOCK÷TIJD)
"END"
```

```
'BEGIN' 'INT' N=4999,SQRTN=25;
    'INT' I:=0,X:=1,LIM:=1,SQUARE:=4,PRX:=1,FX,MAXFX:=0;
    'REAL' TIJD;
    [1:SQRTN] 'INT' LAST,P,V;
    [1:10] 'INT' UIT1,UIT2;
    'PROC' UITVOER=('INT' I) 'VOID';
    'BEGIN' 'FOR' J 'TO' I
    'DO' PRINT((WHOLE(UIT1[J],8),WHOLE(UIT2[J],4))) 'OD';
     PRINT(NEWLINE) 'END';
    'PROC' PRIEM=('REF' 'INT' X) 'VOID';
    'BEGIN' 'BOOL' PRIME:='FALSE';
        'WHILE' 'NOT' PRIME
        'DO' X:=X+2;
             'IF' SQUARE<=X 'THEN' V[LIM]:=SQUARE; LIM+:=1;
                  SQUARE:=P[LIM]**2
             'FI';
             PRIME:='TRUE';
             'FOR' K 'FROM' 2 'TO' LIM 'WHILE' PRIME
             'DO' 'IF' V[K]<X 'THEN' V[K]+:=P[K] 'FI';
                  PRIME:=X 'NE' V[K]
             'OD'
        'OD'
    'END';
    'PROC' F=('INT' P) 'INT';
    'BEGIN' [1:P] 'BOOL' J;
        'INT' A:=0,B,Q:=P 'OVER' 2,U,TL;
        'FOR' I 'TO' P-1 'DO' J[I]:='FALSE' 'OD';
        'FOR' U 'BY' 2 'TO' P-2
        'DO' A+:=U; B:=A+P; 'IF' B>0 'THEN' A:=B 'FI'; J[A]:='TRUE'
        'OD';
         U:=2; 'WHILE' J[U] 'DO' U:=U+1 'OD';
        'IF' J[P-1] 'THEN' TL:=2*U-1 'ELSE' TL:=U 'FI';
        'IF' U>Q-TL 'AND' U<=Q 'THEN' U:=Q 'ELSE' U+:=TL 'FI';
        'WHILE' U<=Q
        'DO' 'BOOL' JA:=J[U]; A:=U-1; B:=U+1;
             'WHILE' J[A]=JA 'DO' A:=A-1 'OD';
             'WHILE' J[B]=JA 'DO' B:=B+1 'OD';
             'IF' B-A-1>TL 'THEN' TL:=B-A-1 'FI';
             'IF' B>Q-TL 'AND' B<=Q 'THEN' U:=Q 'ELSE' U:=B+TL 'FI'
        'OD';
        TL
    'END';

    TIJD:=CLOCK; PRINT((NEWPAGE,"TIJD=",FIXED(TIJD,8,3),NEWLINE));
    'FOR' I 'TO' SQRTN 'DO' LAST[I]:=0 'OD';
    P[1]:=2; I:=0;
    'WHILE' X<N
    'DO' PRIEM(X); I+:=1; UIT1[I]:=X; FX:=F(X); UIT2[I]:=FX; PRX+:=1;
         'IF' PRX<SQRTN 'THEN' P[PRX]:=X 'FI';
         'IF' X>LAST[FX] 'THEN' LAST[FX]:=X 'FI';
         'IF' FX>MAXFX 'THEN' MAXFX:=FX 'FI';
         'IF' I=10 'THEN' UITVOER(I); I:=0 'FI'
    'OD';
    UITVOER(I);
    PRINT((NEWPAGE,"LIJST VAN PRIEMGETALLEN WAARVOOR F HET LAATST DE ",
           "WAARDE N AANNAM",NEWLINE));
    'FOR' I 'TO' MAXFX
    'DO' PRINT((WHOLE(I,3),WHOLE(LAST[I],10),NEWLINE)) 'OD';
   PRINT((NEWLINE,"VERBRUIKTE TIJD",FIXED(CLOCK-TIJD,10,3)))
'END'
```

```
      PROGRAM PRIEMEN(OUTPUT)
      EXTERNAL IF
      COMMON /IFCON/J $ INTEGER J(4999)
      COMMON /COMPR/P,V,SQUARE,LIM
      INTEGER P(25),V(25),SQUARE,LIM
      INTEGER LAST(25),UIT1(10),UIT2(10)
      INTEGER N,SQRTN,I,X,PRX,FX,MAXFX
      REAL TIJD
      DATA (N=4999),(SQRTN=25)
      TIJD=SECOND(CP)
      PRINT 10,TIJD
10    FORMAT (*1TIJD=*,F8.3)
      DO 20 I=1,SQRTN
20    LAST(I)=0
      X=1 $ LIM=1 $ SQUARE=4 $ I=0 $ P(1)=2 $ PRX=1 $ MAXFX=0
30    CALL PRIEM(X) $ I=I+1 $ UIT1(I)=X $ FX=IF(X) $ UIT2(I)=FX
      PRX=PRX+1 $ IF (PRX .LT. SQRTN) P(PRX)=X
      IF (X .GT. LAST(FX)) LAST(FX)=X
      IF (FX .GT. MAXFX) MAXFX=FX
      IF (I .NE. 10) GOTO 50
      PRINT 45,(UIT1(I),UIT2(I),I=1,10)
      I=0
45    FORMAT(1H ,10(I8,I4))
50    IF (X .LT. N) GOTO 30
      PRINT 45,(UIT1(I1),UIT2(I1),I1=1,I)
      PRINT 60
60    FORMAT(*1LIJST VAN PRIEMGETALLEN WAARVOOR F HET LAATST DE*,
     $ * WAARDE N AANNAM*)
      DO 70 X=1,MAXFX
70    PRINT 75,X,LAST(X)
75    FORMAT(1H ,I3,I10)
      PRINT 85,SECOND(CP)-TIJD
85    FORMAT(*0VERBRUIKTE TIJD*,F8.3)
      END
      FUNCTION IF(P) $ INTEGER P
      INTEGER TL,A,B,U,Q,PM1,JA
      COMMON /IFCON/J $ INTEGER J(4999)
      PM1=P-1
      DO 10 A=2,PM1
10    J(A)=0
      A=0 $ U=1
20    IF (U .GT. P-2) GOTO 30
      A=A+U $ U=U+2 $ B=A-P
      IF (B .GT. 0) A=B $ J(A)=1
      GOTO 20
30    Q=PM1/2 $ U=2
40    IF (J(U) .EQ. 0) GOTO 45 $ U=U+1 $ GOTO 40
45    IF (J(PM1) .EQ. 1) 41,42
41    TL=2*U-1 $ GOTO 50
42    TL=U
50    IF ((U .GT. Q-TL) .AND. (U .LE. Q)) 51,52
51    U=Q $ GOTO 60
52    U=U+TL
60    IF (U .GT. Q) GOTO 100
      JA=J(U) $ A=U-1 $ B=U+1
70    IF (J(A) .NE. JA) GOTO 80 $ A=A-1 $ GOTO 70
80    IF (J(B) .NE. JA) GOTO 90 $ B=B+1 $ GOTO 80
90    IF (B-A-1 .GT. TL) TL=B-A-1
      IF ((B .GT. Q-TL) .AND. (B .LE. Q)) 96,97
96    U=Q $ GOTO 60
97    U=B+TL $ GOTO 60
100   IF=TL
      END
      SUBROUTINE PRIEM(X) $ INTEGER X
      COMMON /COMPR/P,V,SQUARE,LIM
      INTEGER P(25),V(25),SQUARE,LIM
      INTEGER K $ LOGICAL NPRIM
10    X=X+2
      IF (SQUARE .GT. X)  GOTO 30
      V(LIM)=SQUARE $ LIM=LIM+1 $ SQUARE=P(LIM)**2
30    K=2 $ NPRIM=.F.
40    IF (NPRIM .OR. (K .GE. LIM)) GOTO 60
      IF (V(K) .LT. X) V(K)=V(K)+P(K)
      NPRIM=(X .EQ. V(K)) $ K=K+1 $ GOTO 40
60    IF (NPRIM) GOTO 10
      RETURN
      END
```

```
PROGRAM PRIEMFUNCTIE (OUTPUT);
    CONST N=4999; SQRTN=25;
    VAR   LAST,P,V: ARRAY[1:SQRTN] OF INTEGER;
          UIT1,UIT2: ARRAY[1:10] OF INTEGER;
          I,X,LIM,SQUARE,PRX,FX,MAXFX: INTEGER;
          TIJD: REAL;

    FUNCTION F(P: INTEGER): INTEGER;
        VAR TL,A,B,U,Q: INTEGER;
            JA: BOOLEAN;
            J: ARRAY[1:N] OF BOOLEAN;
    BEGIN FOR A:=2 TO P-1 DO J[A]:=FALSE;
       A:=0; U:=1;
       WHILE U<=P-2 DO
          BEGIN A:=A+U; U:=U+2;
            B:=A+P; IF B>0 THEN A:=B; J[A]:=TRUE
          END;
       Q:=P DIV 2; U:=2;
       WHILE J[U] DO U:=U+1;
       IF J[P-1] THEN TL:=2*U-1 ELSE TL:=U;
       IF (U>Q-TL) AND (U<=Q) THEN U:=Q ELSE U:=U+TL;
       WHILE U<=Q DO
       BEGIN JA:=J[U]; A:=U-1; B:=U+1;
             WHILE J[A]=JA DO A:=A-1;
             WHILE J[B]=JA DO B:=B+1;
             IF B-A-1>TL THEN TL:=B-A-1;
             IF (B>Q-TL) AND (B<=Q) THEN U:=Q ELSE U:=B+TL
       END;
       F:=TL
    END;

    PROCEDURE UITVOER(N: INTEGER);
        VAR I: INTEGER;
    BEGIN FOR I:=1 TO N DO WRITE(UIT1[I]:8,UIT2[I]:4); WRITELN
    END;

    PROCEDURE PRIEM(VAR X: INTEGER);
        VAR K: INTEGER;
            PRIM: BOOLEAN;
    BEGIN REPEAT X:=X+2;
          IF SQUARE<=X THEN
          BEGIN V[LIM]:=SQUARE; LIM:=LIM+1; SQUARE:=SQR(P[LIM]) END;
          K:=2; PRIM:=TRUE;
          WHILE PRIM AND (K<LIM) DO
              BEGIN IF V[K]<X THEN V[K]:=V[K]+P[K];
                    PRIM:=(X<>V[K]); K:=K+1
              END;
      UNTIL PRIM;
   END;

BEGIN TIJD:=CLOCK; WRITELN('1TIJD=',TIJD/1000:8:3);
      FOR I:=1 TO SQRTN DO LAST[I]:=0; MAXFX:=0;
      X:=1; LIM:=1; SQUARE:=4; I:=0; P[1]:=2; PRX:=1;
      REPEAT PRIEM(X); I:=I+1; UIT1[I]:=X; FX:=F(X); UIT2[I]:=FX;
             PRX:=PRX+1; IF PRX<SQRTN THEN P[PRX]:=X;
             IF X>LAST[FX] THEN LAST[FX]:=X;
             IF FX>MAXFX THEN MAXFX:=FX;
             IF I=10 THEN BEGIN UITVOER(I); I:=0 END;
      UNTIL X>=N;
      UITVOER(I);
      WRITELN('1LIJST VAN PRIEMGETALLEN WAARVOOR F HET LAATST DE');
      WRITELN(' WAARDE N AANNAM'); WRITELN;
      FOR X:=1 TO MAXFX DO WRITELN(X:3,LAST[X]:10);
      WRITELN('0VERBRUIKTE TIJD',(CLOCK-TIJD)/1000:8:3)
END.
```

```
"BEGIN"  "REAL"  TIME;
 "PROCEDURE" PERFECT SQUARED RECTANGLE(M,N); "VALUE" M,N; "INTEGER" M,N;
   "BEGIN" "INTEGER" AANTAL; "BOOLEAN" GEEN OPLOSSING;
       "COMMENT"  "CHECKON"  VIERK;
       "INTEGER" "ARRAY" VIERK[1:M];


       "PROCEDURE"  Z O E K   V I E R K A N T E N ;

        "BEGIN" "INTEGER" NIV,COMBNR,K;
                "COMMENT" "CHECKON" S; "INTEGER" "ARRAY" S[0:M];


          "PROCEDURE" SPLITS(GETAL,VANAF); "VALUE" GETAL,VANAF;
                                           "INTEGER" GETAL,VANAF;
           "BEGIN" "INTEGER" I;
             "FOR" I:= VANAF "STEP" -1 "UNTIL" 1 "DO"
             "BEGIN" "INTEGER" NIEUW; NIEUW:= GETAL-I*I;
               "IF" NIEUW <= S[I-1] "THEN"
               "BEGIN"
                 "IF" NIEUW > 0 "THEN"
                 "BEGIN" "INTEGER" VAN,DUM;
                   VIERK[NIV]:= I; VAN:= I-1;
                   "FOR" DUM:= 0 "WHILE" VAN*VAN>NIEUW "DO" VAN:= VAN-1;
                   "IF" NIV=1 & VAN+I>N "THEN" VAN:= N-I;
                   NIV:= NIV+1; SPLITS(NIEUW,VAN); NIV:= NIV-1
                 "END"
                 "ELSE"
                 "IF" NIEUW=0 "THEN"
                 "BEGIN" VIERK[NIV]:= I; AANTAL:= NIV; COMBNR:=COMBNR+1;
                   "IF" AANTAL >= 9  &  COMBNR>=40  &  COMBNR<=90
                   "THEN"  P R O B E R E N
                 "END"
               "END"
               "ELSE" I:= 1
             "END"   DO LOOP
           "END"  SPLITS;

          NIV:= 1; COMBNR:= 0; S[0]:= 0;
          "FOR" K:= 1 "STEP" 1 "UNTIL" M "DO" S[K]:= S[K-1]+K*K;
          SPLITS(N*M,M)
        "END"  ZOEK VIERKANTEN;


       "PROCEDURE"  P R O B E R E N ;

        "BEGIN" "INTEGER" MP1,MP1,REGNR,VOLGNR,LINKSBOVEN,K;
          "COMMENT"  "CHECKON" GEBRUIKT, RAND,HOR,VERT,X,Y,R;
          "BOOLEAN" "ARRAY" GEBRUIKT, RAND[1:AANTAL];
          "INTEGER" "ARRAY" HOR[1:M+1],VERT[1:N],X,Y,R[1:AANTAL];


          "PROCEDURE" LEG(XVRY,YVRY); "VALUE" XVRY,YVRY;
                                      "INTEGER" XVRY,YVRY;
           "BEGIN" "INTEGER" GAT,I,NR;  GAT:= 0;
             "FOR" I:= XVRY "STEP" 1 "UNTIL" N "DO"
             "IF" VERT[I]=YVRY "THEN" GAT:= GAT+1 "ELSE" I:= N;
             "IF" GAT>MP1-YVRY "THEN" GAT:= MP1-YVRY;
             "FOR" NR:= 1 "STEP" 1 "UNTIL" AANTAL "DO"
             "IF" ~GEBRUIKT[NR] & VIERK[NR]<=GAT "THEN"
             "BEGIN" "INTEGER" RIBBE,RB,LO;
               "BOOLEAN" VERT RAND, HOR RAND, HOEK, BINNEN, MOGELYK;
               RIBBE:= VIERK[NR]; RB:= XVRY+RIBBE-1; LO:= YVRY+RIBBE-1;
               VERTRAND:= XVRY=1 "OR" RB=N; HORRAND:= YVRY=1 "OR" LO=M;
               HOEK  :=  VERT RAND  &  HOR RAND;
               BINNEN:= ~VERT RAND  &  ~HOR RAND;
               "IF" VOLGNR=1 "THEN" LINKSBOVEN:= RIBBE;
               MOGELYK:=  BINNEN   "OR"   ( ~HOEK  &  RAND[NR] )  "OR"
                        ( HOEK  &  RIBBE<=LINKSBOVEN  &  RAND[NR] );
               "IF"  MOGELYK "THEN"
               "BEGIN" "INTEGER" K,DUM; "BOOLEAN" VOL;
                 "CHECKON" HULP;
                 "INTEGER" "ARRAY" HULP[YVRY:LO];
                 GEBRUIKT[NR]:= "TRUE";
                 X[VOLGNR]:= XVRY; Y[VOLGNR]:= YVRY; R[VOLGNR]:= RIBBE;
                 "FOR" K:=YVRY "STEP" 1 "UNTIL" LO "DO" HULP[K]:=HOR[K];
                 "FOR" K:= YVRY "STEP" 1 "UNTIL" LO "DO"
                 "IF" HOR[K] = XVRY "THEN" HOR[K]:= HOR[K]+RIBBE
                    "ELSE" K:= LO ;
                 "FOR" K:= XVRY "STEP" 1 "UNTIL" RB "DO"
```

```
                VERT[K]:= VERT[K]+RIBBE;
            VOL:= "TRUE";
            "FOR" DUM:= 0 "WHILE" VOL "DO"
            "BEGIN"
              "FOR" K:= HOR[REGNR] "STEP" 1 "UNTIL" N "DO"
              "IF" VERT[K]=REGNR "THEN" HOR[REGNR]:= HOR[REGNR]+1
                 "ELSE" K:= N ;
              "IF" HOR[REGNR]=NP1 "THEN" REGNR:= REGNR+1
                 "ELSE" VOL:= "FALSE"
            "END";
            "IF" REGNR=NP1 "THEN"
            "BEGIN" GEEN OPLOSSING:= "FALSE";  C O D E  "END"
            "ELSE"
            "BEGIN"  VOLGNR:= VOLGNR+1; LEG(HOR[REGNR],REGNR);
                     VOLGNR:= VOLGNR-1
            "END";
            REGNR:= YVRY; GEBRUIKT[NR]:= "FALSE";
            "FOR" K:=YVRY "STEP" 1 "UNTIL" LO "DO" HOR[K]:=HULP[K];
            "FOR" K:=XVRY "STEP" 1 "UNTIL" RB "DO"
               VERT[K]:= VERT[K]-RIBBE
          "END" MOGELIJK
        "END"
      "END" LEG;


      "PROCEDURE"  C O D E ;

       "BEGIN" "INTEGER" I;
         OUTPUT(61,"("""("COORDINATEN VAN DE HOEKEN LINKSONDER")",
                   3/")");
         "FOR" I:= 1 "STEP" 1 "UNTIL" AANTAL "DO"
          OUTPUT(61,"("3(9ZD),/")",X[I]-1,Y[I]-1,R[I])
       "END" CODE;

      NP1:= N+1; MP1:= M+1; REGNR:= VOLGNR:= 1;
      "FOR" K:= 1 "STEP" 1 "UNTIL" NP1 "DO" HOR[K]:= 1;
      "FOR" K:= 1 "STEP" 1 "UNTIL" N "DO" VERT[K]:= 1;
      "FOR" K:= 1 "STEP" 1 "UNTIL" AANTAL "DO"
      "BEGIN" GEBRUIKT[K]:= "FALSE"; RAND[K]:= "TRUE" "END";
      RAND[AANTAL]:= RAND[AANTAL-1]:= RAND[AANTAL-2]:= "FALSE";
      L E G (1,1)
    "END" PROBEREN;


  OUTPUT(61,"("""(" RECHTHOEK  VAN ")",9ZD,/,
                "("             BIJ ")",9ZD,3/")",M,N);
  GEEN OPLOSSING:= "TRUE";   Z O E K   V I E R K A N T E N ;
  "IF" GEEN OPLOSSING "THEN"
   OUTPUT(61,"("""(" BEZIT GEEN PERFECTE VERDELING")",/")")

"END"  PERF. SQUARED RECT.;

TIME:= CLOCK;
PERFECT SQUARED RECTANGLE(32,33);
TIME:= CLOCK-TIME; OUTPUT(61,"("-")",TIME)
"END"  OF PROGRAM
```

```
(   'REAL' TIME;
'PROC'  PERFECT SQUARED RECTANGLE = ( 'INT' M, N ) 'VOID' ;

 ( 'INT' AANTAL; 'BOOL' GEEN OPLOSSING:= 'TRUE'; [1:M]'INT' VIERK;


   'PROC'  ZOEK VIERKANTEN = 'VOID' ;

    (  'INT' NIV:= 1; [0:M]'INT' S;
       'INT' COMBNR:= 0;


      'PROC'  SPLITS = ('INT' GETAL,VANAF) 'VOID' ;

       (  'BOOL' POSS:='TRUE';
         'FOR' I 'FROM' VANAF 'BY' -1 'TO' 1 'WHILE' POSS
         'DO'  'INT' NIEUW=GETAL-I*I;
            'IF' NIEUW <= S[I-1] 'THEN'
               'IF' NIEUW > 0 'THEN'
                   VIERK[NIV]:= I;
                   'INT' VAN:=I-1;
                   'WHILE' VAN*VAN > NIEUW 'DO' VAN-:=1 'OD';
                   'IF' NIV=1 'AND' VAN+I > N 'THEN' VAN:= N-I 'FI';
                   NIV+:= 1; SPLITS(NIEUW,VAN); NIV-:= 1
               'ELIF' NIEUW=0 'THEN'
                   VIERK[NIV]:= I; AANTAL:= NIV;
                   COMBNR+:= 1;
                   'IF' AANTAL>=9 'AND' COMBNR>=40 'AND' COMBNR<=90
                      'THEN' PROBEREN
                   'FI'
               'FI'
            'ELSE' POSS:= 'FALSE'
            'FI'
         'OD'
       );


       S[0]:=0; 'FOR' I 'TO' M 'DO' S[I]:=S[I-1]+I*I 'OD';
       SPLITS(N*M,M)
     );                    # EINDE PROC ZOEKVIERKANTEN #


   'PROC'  PROBEREN = 'VOID' ;

    (  'INT' NP1=N+1, MP1=M+1; 'INT' REGNR:= 1, VOLGNR:= 1;
       'INT' LINKSBOVEN, [1:AANTAL]'BOOL' GEBRUIKT, RAND;
       [1:MP1]'INT' HOR, [1:N]'INT' VERT; [1:AANTAL]'INT' X, Y, R;


      'PROC'  LEG = ('INT' XVRY,YVRY) 'VOID' ;

       (  'INT' GAT:= 0;
         'FOR' I 'FROM' XVRY 'TO' N 'WHILE' VERT[I]=YVRY
         'DO' GAT+:= 1 'OD';
         'IF' GAT > MP1-YVRY 'THEN' GAT:=MP1-YVRY 'FI';
         'FOR' NR 'TO' AANTAL 'DO'
         'IF' 'NOT' GEBRUIKT[NR] 'AND' VIERK[NR]<=GAT 'THEN'

            'INT' RIBBE=VIERK[NR];
            'INT' RB=XVRY+RIBBE-1, LO=YVRY+RIBBE-1;#RECHTSBOV,LINKSOND#
            'BOOL'  VERT RAND= XVRY=1 'OR' RB=N,
                     HOR RAND= YVRY=1 'OR' LO=M;
            'BOOL' HOEK= VERT RAND 'AND' HOR RAND,
                 BINNEN= 'NOT' VERT RAND  'AND'  'NOT' HOR RAND;
            'IF' VOLGNR=1 'THEN' LINKSBOVEN:= RIBBE 'FI';
            'BOOL' MOGELYK= BINNEN 'OR' ('NOT'HOEK 'AND' RAND[NR]) 'OR'
                          (HOEK 'AND' RIBBE<=LINKSBOVEN 'AND' RAND[NR]);

            'IF' MOGELYK 'THEN'
                GEBRUIKT[NR]:= 'TRUE';
                X[VOLGNR]:= XVRY; Y[VOLGNR]:= YVRY; R[VOLGNR]:= RIBBE;
                [1:RIBBE]'INT' HULP;
                'FOR' II 'TO' RIBBE 'DO' HULP[II]:= HOR[YVRY+II-1]'OD';
                'FOR' I 'FROM' YVRY 'TO' LO 'WHILE' HOR[I]=XVRY
                'DO' HOR[I]+:= RIBBE 'OD';
                'FOR' I 'FROM' XVRY 'TO' RB 'DO' VERT[I]+:= RIBBE 'OD';
                'BOOL' VOL:= 'TRUE';
                'WHILE' VOL 'DO'
                   'FOR' I 'FROM' HOR[REGNR] 'TO' N
                   'WHILE' VERT[I] 'NE' REGNR 'DO' HOR[REGNR]+:=1 'OD';
```

```
                    'IF' HOR[REGNR]=MP1 'THEN' REGNR+:= 1
                                        'ELSE' VOL:='FALSE'
                    'FI'
                'OD';
                'IF' REGNR = MP1 'THEN'
                    GEEN OPLOSSING:= 'FALSE';   C O D E
                'ELSE' VOLGNR+:=1; LEG(HOR[REGNR],REGNR); VOLGNR:=1
                'FI';
                REGNR:= YVRY;
                'FOR' II 'FROM' YVRY 'TO' LO 'DO'
                HOR[II]:= HULP[II-YVRY+1] 'OD';
                'FOR' I 'FROM' XVRY 'TO' RB 'DO' VERT[I]:= RIBBE 'OD';
                GEBRUIKT[NR]:= 'FALSE'
            'FI'
        'FI'
        'OD'
    );

    'PROC'  CODE = 'VOID' ;

     (  PRINT(("COORDINATEN VAN DE HOEKEN LINKSONDER",
               NEWLINE,NEWLINE,NEWLINE) );
     'FOR' I 'TO' AANTAL 'DO' PRINT((X[I]-1,Y[I]-1,R[I],NEWLINE))'OD'
     );

    'FOR' I 'TO' MP1 'DO' HOR[I]:= 1 'OD';
    'FOR' I 'TO' N 'DO' VERT[I]:= 1 'OD';
    'FOR' I 'TO' AANTAL
    'DO'  GEBRUIKT[I]:= 'FALSE'; RAND[I]:= 'TRUE' 'OD';
     RAND[AANTAL-2]:= RAND[AANTAL-1]:= RAND[AANTAL]:= 'FALSE';
     LEG(1,1)
  );              # EINDE PROC PROBEREN #

  PRINT((NEWPAGE," RECHTHOEK VAN ",M,NEWLINE,
                 "          BIJ ",N,NEWLINE,NEWLINE,NEWLINE));
  Z O E K  V I E R K A N T E N;
  'IF' GEEN OPLOSSING
  'THEN' PRINT( (" BEZIT GEEN PERFECTE VERDELING",NEWLINE) )
  'FI'
);                    # EINDE PROC PERF. SQUARED RECT. #

TIME:= CLOCK;
P E R F E C T  S Q U A R E D  R E C T A N G L E (32,33);
TIME:= CLOCK-TIME; PRINT( (NEWPAGE,TIME) )
)
```

```
PROGRAM RECTANGLE(OUTPUT);

CONST   M = 32;   N = 33;  (*  EIS; M <= N   *)
      MP1 = 33; NP1 = 34;

VAR   TIJD : REAL;

PROCEDURE   PERFECTSQUAREDRECTANGLE ;
   VAR   AANTAL  : 1..M;
         GEENOPL : BOOLEAN;
         VIERK   : ARRAY [1..M] OF INTEGER;

  PROCEDURE   PROBEREN ;  FORWARD;

   PROCEDURE   ZOEKVIERKANTEN ;
      VAR   NIV, COMBNR, K : INTEGER;
            S              : ARRAY [0..M] OF INTEGER;

      PROCEDURE   SPLITS( GETAL, VANAF : INTEGER );
         VAR   I, NIEUW, VAN : INTEGER;
      BEGIN   I:= VANAF;
         WHILE  I>=1  DO
         BEGIN   NIEUW:= GETAL-SQR(I);
            IF NIEUW<=S[I-1] THEN
            BEGIN
               IF NIEUW>0 THEN
               BEGIN   VIERK[NIV]:= I; VAN:= I-1;
                  WHILE SQR(VAN)>NIEUW DO VAN:= VAN-1;
                  IF (NIV=1) AND (VAN+I>N) THEN VAN:= N-I;
                  NIV:= NIV+1; SPLITS(NIEUW,VAN); NIV:= NIV-1
               END
               ELSE
               IF NIEUW=0 THEN
               BEGIN   VIERK[NIV]:= I; AANTAL:= NIV; COMBNR:= COMBNR+1;
                  IF (AANTAL>=9) AND (COMBNR>=40) AND (COMBNR<=90)
                  THEN   PROBEREN
               END
            END
            ELSE  I:= 1;
            I:= I-1
         END
      END;

   BEGIN   S[0]:= 0; FOR K:= 1 TO M DO S[K]:= S[K-1]+SQR(K);
           NIV:= 1; COMBNR:= 0;
           SPLITS(N*M,M)
   END;


   PROCEDURE   PROBEREN ;
      VAR  REGNR, VOLGNR, LINKSBOVEN, K : INTEGER;
           GEBRUIKT, RAND               : ARRAY [1..M] OF BOOLEAN;
           HOR                          : ARRAY [1..MP1] OF INTEGER;
           VERT                         : ARRAY [1..N] OF INTEGER;
           X, Y, R                      : ARRAY [1..M] OF INTEGER;

      PROCEDURE   CODE ;
         VAR  I : INTEGER;
      BEGIN   WRITELN(' COORDINATEN VAN DE HOEKEN LINKSONDER');
         WRITELN; WRITELN;
         FOR I:= 1 TO AANTAL DO WRITELN( X[I]-1, Y[I]-1, R[I])
      END;

      PROCEDURE   LEG( XVRY, YVRY : INTEGER) ;
         VAR  GAT, NR, I  : INTEGER ;

         PROCEDURE  MISSCHIEN ;
            VAR  RIBBE, RB, LO, I                          : INTEGER;
                 HORRAND, VERTRAND, HOEK, BINNEN, MOGELYK : BOOLEAN;

            PROCEDURE  LEGTERUGHAALWEG ;
               VAR  K   : INTEGER;
                    VOL : BOOLEAN;
                   HULP : ARRAY [1..MP1] OF INTEGER;
            BEGIN  GEBRUIKT[NR]:= TRUE;
               X[VOLGNR]:= XVRY; Y[VOLGNR]:= YVRY; R[VOLGNR]:= RIBBE;
               FOR K:= YVRY TO LO DO HULP[K]:= HOR[K];
               K:= YVRY;
               WHILE  K<=LO DO
               BEGIN  IF HOR[K]=XVRY THEN HOR[K]:= HOR[K]+RIBBE
```

```
                    ELSE  K:= LO;
                 K:= K+1
              END;
              FOR K:= XVRY TO RB DO  VERT[K]:= VERT[K]+RIBBE;
              VOL:= TRUE;
              WHILE  VOL  DO
              BEGIN  K:= HOR[REGNR];
                 WHILE  K<=N  DO
                 BEGIN IF VERT[K]<>REGNR THEN HOR[REGNR]:= HOR[REGNR]+1
                       ELSE  K:= N;
                    K:= K+1
                 END;
                 IF HOR[REGNR]=NP1 THEN REGNR:= REGNR+1
                 ELSE  VOL:= FALSE
              END;
              IF  REGNR=NP1  THEN BEGIN GEENOPL:= FALSE; CODE END
              ELSE
              BEGIN  VOLGNR:= VOLGNR+1; LEG(HOR[REGNR],REGNR);
                     VOLGNR:= VOLGNR-1
              END;
              REGNR:= YVRY; GEBRUIKT[NR]:= FALSE;
              FOR K:= YVRY TO LO DO HOR[K]:= HULP[K];
              FOR K:= XVRY TO RB DO VERT[K]:= VERT[K]-RIBBE
           END;

        BEGIN  RIBBE:= VIERK[NR]; RB:= XVRY+RIBBE-1; LO:= YVRY+RIBBE-1;
           HORRAND:= (YVRY=1) OR (LO=M); VERTRAND:= (XVRY=1) OR (RB=N);
           HOEK   :=      HORRAND  AND       VERTRAND;
           BINNEN:= NOT HORRAND  AND  NOT VERTRAND;
           IF  VOLGNR=1 THEN LINKSBOVEN:= RIBBE;
           MOGELYK:=  BINNEN  OR  ( NOT HOEK  AND  RAND[NR] )  OR
                      ( HOEK  AND  (RIBBE<=LINKSBOVEN)  AND  RAND[NR]);
           IF  MOGELYK  THEN  LEGNEERHAALNEG
        END;

     BEGIN  GAT:= 0; I:= XVRY;
        WHILE  I<=N  DO
        BEGIN  IF  VERT[I]=YVRY  THEN  GAT:= GAT+1  ELSE  I:= N;
           I:= I+1
        END;
        IF  GAT>NP1-YVRY  THEN  GAT:= NP1-YVRY;
        FOR NR:= 1 TO AANTAL DO
        IF  NOT GEBRUIKT[NR]  AND  (VIERK[NR]<=GAT) THEN  MISSCHIEN
     END;

  BEGIN   REGNR:= 1; VOLGNR:= 1;
     FOR K:= 1 TO NP1 DO HOR[K]:= 1;
     FOR K:= 1 TO N DO VERT[K]:= 1;
     FOR K:= 1 TO AANTAL DO
     BEGIN  GEBRUIKT[K]:= FALSE; RAND[K]:= TRUE  END;
     RAND[AANTAL-2]:= FALSE; RAND[AANTAL-1]:= FALSE;
     RAND[AANTAL]:= FALSE;
     LEG( 1, 1 )
  END;

BEGIN   WRITELN('RECHTHOEK VAN',M);
        WRITELN('          BIJ',N);
  WRITELN; WRITELN;
  GEENOPL:= TRUE;  ZOEKVIERKANTEN;
  IF GEENOPL THEN WRITELN(' BEZIT GEEN PERFECTE VERDELING')
END;

BEGIN   TIJD:= CLOCK;
  PERFECTSQUAREDRECTANGLE; TIJD:= CLOCK-TIJD;
  WRITE('1',TIJD)
END.
```

```
"BEGIN" "INTEGER" I, J, K;
    "REAL" P1, X;
    P1:= CLOCK;
    "FOR" I:= 1 "STEP" 1 "UNTIL" 1000 "DO"
    "FOR" J:= 1 "STEP" 1 "UNTIL" I "DO"
    "BEGIN" X:= (I*J)/I;
        "IF" X =/= J "THEN"
        "BEGIN" K:= K + 1;
            OUTPUT(61,"("""("ONJUIST UITGEVOERDE DELING")",6ZD,
            "(" / ")",4ZD,"(" =")",N,"(" IN PLAATS VAN")",4ZD,/")",
            I*J,I,X,J)
        "END"
    "END";
    OUTPUT(61,"("""("AANTAL ONJUIST UITGEVOERDE DELINGEN:")",5ZD,/")",K);
    OUTPUT(61,"("""("REKENTIJD: ")",3ZD.3D,/")",CLOCK-P1)
"END"


'BEGIN'
    'INT' K:=0; 'REAL' P1=CLOCK;
    'FOR' I 'TO' 1000
    'DO' 'FOR' J 'TO' I
         'DO' 'REAL' X=(I*J)/I;
              'IF' X/=J 'THEN' PRINT(("ONJUIST UITGEVOERDE DELING:",
                   WHOLE(I*J,0)," /",WHOLE(I,0)," =",X,
                   "IN PLAATS VAN", WHOLE(J ,0),NEWLINE))
              'FI'
         'OD'
    'OD';
    PRINT(("AANTAL ONJUIST UITGEVOERDE DELINGEN: ",WHOLE(K,0),NEWLINE));
    PRINT(("REKENTIJD: ",FIXED(CLOCK-P1,10,3),NEWLINE))
'END'


      PROGRAM DTEST(OUTPUT)
      P1=SECOND(CP)
      K=0
      DO 66 I=1,1000
      DO 66 J=1,I
      X=(I*J)/I
      IF (X-J) 65,66
   65 PRINT 70,I*J,I,X,J
   70 FORMAT(* ONJUIST UITGEVOERDE DELING: *,I8,* /*,I6,* =*,F20.15,
     C * IN PLAATS VAN*,I6)
      K=K+1
   66 CONTINUE
      PRINT 71,K
   71 FORMAT(* AANTAL ONJUIST UITGEVOERDE DELINGEN: *,I8)
      PRINT 72,SECOND(CP)-P1
   72 FORMAT(* REKENTIJD: *,F6.3)
      END


PROGRAM DTEST(OUTPUT);
VAR I,J,K:INTEGER;X,P1,P2:REAL;
BEGIN
P1:=CLOCK;
FOR I:=1 TO 1000 DO
FOR J:=1 TO I DO
BEGIN
X:=(I*J)/I;
IF X<>J THEN
  BEGIN K:=K+1;
   WRITELN(' ONJUIST UITGEVOERDE DELING: ',I*J:8,' / ',I:6,
     ' = ',X,' IN PLAATS VAN ',J:6)
  END
END
WRITELN(' AANTAL ONJUIST UITGEVOERDE DELINGEN : ', K:6);
P2:=CLOCK;
WRITELN(' REKENTIJD : ', P2-P1)
END.
```

```
"BEGIN""COMMENT" JKOK, 750604, PRINT OPLOSSINGEN VAN 15 * 4 PENTOMINO
      PROBLEEM;

  "PROCEDURE" PRINT OPL;
  "BEGIN"
     "PROCEDURE" LIST OPL(ITEM); "PROCEDURE" ITEM;
     "BEGIN""INTEGER" I, J, A, B, C, D, MAX;
        PAG:= PAG + 1; ITEM(LAST + 1); ITEM(AANTAL); ITEM(PAG);
        MAX:= (AANTAL - LAST + 1) // 2 * 5 - 1;
        "FOR" I:= 0 "STEP" 1 "UNTIL" MAX "DO"
        "BEGIN""FOR" J:= 0 "STEP" 1 "UNTIL" 31 "DO"
           "BEGIN" A:= X[I,J]; B:= X[I,J + 1]; C:= X[I + 1, J];
              D:= X[I + 1, J + 1];
              "IF" A = C "THEN"
              "BEGIN""IF" B = D "THEN"
                 "BEGIN""IF" A = B "THEN" ITEM("("   ")") "ELSE"
                    ITEM("("  |")")
                 "END""ELSE" ITEM("("  +")")
              "END""ELSE"
              "IF" A = B "AND" C = D "THEN" ITEM("("---")") "ELSE"
              ITEM("("--+")")
           "END";
           "FOR" J:= 0 "STEP" 1 "UNTIL" 31 "DO""IF" X[I + 1, J] =
              X[I + 1, J + 1] "THEN" ITEM("("   ")") "ELSE"
                                     ITEM("("  |")")
        "END" FOR I
     "END" LIST OPL;

     "PROCEDURE" LAYOUT;
     FORMAT("("*, "(" NUMMERS ")"3ZD, "(" TOT ")"3ZD, 60B"("BLZ, ")",
        3ZD, X(/, 10(/, 2(10B, 16(3S))))")", (AANTAL - LAST + 1) // 2);

     OUTLIST(2, LAYOUT, LIST OPL); LAST:= AANTAL
  "END" PRINT OPL;

  "PROCEDURE" LEES OPL;
  "BEGIN""PROCEDURE" LAYIN; FORMAT("("B, 60(A)")");

     "PROCEDURE" LISTIN(ITEM); "PROCEDURE" ITEM;
     "BEGIN" "INTEGER" I, J, H;
        "FOR" I:= 1 "STEP" 1 "UNTIL" 15 "DO"
        "FOR" J:= 1 "STEP" 1 "UNTIL" 4 "DO"
        "BEGIN" ITEM(H); X[VERT + J, HOR + 16 - I]:= H //
           439 80465 11104; "IF" H = 0 "THEN""GO TO" ERROR
        "END"
     "END" LISTIN;

     INLIST(1, LAYIN, LISTIN)
  "END" LEES OPL;

  "INTEGER" I, N, AANTAL, PAG, LAST, HOR, VERT;
  "REAL" KLOK;
  "INTEGER""ARRAY" X[0 : 25, 0 : 32];
  AANTAL:= PAG:= LAST:= 0; "FOR" I:= 0 "STEP" 1 "UNTIL" 25 "DO"
  "FOR" N:= 0 "STEP" 1 "UNTIL" 32 "DO" X[I,N]:= 0;
  EOF(1, ERROR); KLOK:= CLOCK;
  "FOR" N:= 1 "WHILE" "TRUE" "DO"
  "BEGIN""FOR" VERT:= 0 "STEP" 5 "UNTIL" 20 "DO"
     "FOR" HOR:= 0, 16 "DO"
     "BEGIN" LEES OPL; AANTAL:= AANTAL + 1; INPUT(1, "("/")") "END";
     PRINT OPL
  "END";
ERROR :
  N:= AANTAL - LAST; "IF" N > 0 "THEN"
  "BEGIN""IF" N // 2 * 2 = N "THEN"
     "BEGIN" VERT:= (N - 1) // 2 * 5; HOR:= 16;
        "FOR" I:= 1 "STEP" 1 "UNTIL" 15 "DO"
        "FOR" N:= 1 "STEP" 1 "UNTIL" 4 "DO" X[VERT + N, HOR + I]:= 0
     "END";
     PRINT OPL
  "END";
  OUTPUT(61, "("///, "("EXECUTIETIJD : ")", 4ZD,3D8, "("SEC.")", /,
     "("AANTAL OPLOSSINGEN IS ")", 4ZD,/")", CLOCK - KLOK, AANTAL)
"END"
```

```
'BEGIN' 'CO' JKOK, 750812, LIST PENTOMINO-OPLOSSINGEN 'CO'
    'INT' AANTAL := 0, PAG := 0, LAST := 0, N, VERT := 0, HOR := 0;
    [0 : 25, 0 : 32] 'CHAR' X;
    'FILE' AUX1 := STAND IN;

    'PROC' LEESOPL = 'VOID' :
    'BEGIN' 'CHAR' H; GET(AUX1, H);
        'FOR' I 'TO' 15 'DO' 'FOR' J 'TO' 4 'DO'
            X[VERT + J, HOR + 16 - I] := (GET(AUX1, H); H);
            'IF' H = " " 'THEN' 'GOTO' EXIT 'FI'
        'OD' 'OD'; NEWLINE(AUX1)
    'END'; 'CO' LEESOPL 'CO'

    'PROC' PRINTOPL = 'VOID' :
    'BEGIN'

        'PROC' SPATIE = ( 'REF' 'FILE' F, 'INT' N ) 'VOID' :
        'FOR' K 'TO' N 'DO' PUT(F," ") 'OD';

        'CHAR' A, B, C, D; PAG +:= 1;
        PUT( UITV, (NEWPAGE, " NUMMERS ", WHOLE((LAST + 1), -4),
                   "  TOT ", WHOLE(AANTAL, -4)));
        SPATIE(UITV,60);
        PUT( UITV, ( "BLZ. ", WHOLE(PAG,-4)));
        'INT' MAX := (AANTAL - LAST + 1) 'OVER' 2 * 5 - 1;
        'FOR' I 'FROM' 0 'TO' MAX
        'DO' 'IF' I 'OVER' 5 * 5 = I 'THEN' NEWLINE(UITV) 'FI';
            SPATIE(UITV, 10); 'FOR' J 'FROM' 0 'TO' 31
            'DO' A := X[I,J]; B := X[I,J + 1]; C := X[I + 1, J];
                D := X[I + 1, J + 1];
                PUT(UITV,(A = C | (B = D | (A = B|"   "|"  |")|"  +")
                                 |: A = B 'AND' C = D | "---" | "--+"));
                'IF' J = 15 'THEN' SPATIE(JITV, 10) 'FI'
            'OD'; NEWLINE(UITV); SPATIE(UITV,10);
            'FOR' J 'FROM' 0 'TO' 31
            'DO' PUT(UITV,(X[I+1,J] = X[I+1,J+1] | "   " | "  |"));
                'IF' J = 15 'THEN' SPATIE(UITV, 10) 'FI'
            'OD'; NEWLINE(UITV)
        'OD'; 'CO' FOR I 'CO'
        LAST := AANTAL
    'END'; 'CO' PRINTOPL 'CO'

    'FILE' UITV;
    ESTABLISH(UITV, "UITV", STAND OUT CHANNEL, 40, 60, 136);
    'FOR' I 'FROM' 0 'TO' 25
    'DO' 'FOR' J 'FROM' 0 'TO' 32 'DO' X[I,J] := " " 'OD'
    'OD';
    'REAL' KLOK := CLOCK; 'BOOL' EIND:= 'FALSE'; 'WHILE' 'NOT' EIND
    'DO' VERT:= HOR:= 0;
        ON LOGICAL FILE END( AUX1,
           ('REF' 'FILE' FILE ) 'BOOL' : (EIND:= 'TRUE'; LOOPEXIT));
        'WHILE' AANTAL +:= 1; LEESOPL;
            HOR := ( HOR = 0 : 16 | VERT:= VERT + 5; 0);
            VERT < 21
        'DO' 'SKIP' 'OD';
        LOOPEXIT :
        N := AANTAL - LAST;
        'IF' N 'MOD' 2 'NE' 0 'THEN'
            VERT := (N - 1) 'OVER' 2 * 5; HOR := 16;
            'FOR' I 'TO' 15
            'DO' 'FOR' J 'TO' 4 'DO' X[VERT + J,HOR + I]:= " " 'OD' 'OD'
        'FI';
        PRINTOPL
    'OD';
    EXIT :
    PRINT((NEWLINE, "EXECUTIETIJD : ", FIXED((CLOCK - KLOK), 11, 3 ),
         " SEC.", NEWLINE, "AANTAL OPLOSSINGEN IS ",WHOLE(AANTAL,-5)));
    CLOSE(UITV)
'END'
```

```
      PROGRAM LISTPE(INPUT,OUTPUT,UITV,
     1   TAPE1 = INPUT,TAPE2 = OUTPUT,TAPE3 = UITV)
C
C     J KOK, 750807, PRINT PENTOMINO-OPLOSSINGEN IN HET KADER VAN
C     BEPALING EXECUTIE-TIJDEN
C
      INTEGER I,N,AANTAL,PAG,LAST,HOR,VERT,A,B,C,D,PCC,
     1    IX(26,33), UU(32), STR(5)
      REAL KLOK, HULP, EOF1
C     LOWER BOUNDS ZIJN 1
C
      DATA STR/3R   , 3R  1, 3R  +, 3R--+, 3R---/
      AANTAL = 0
      PAG = 0
      LAST = 0
      EOF1 = 0.
      DO 1 I = 1,26
      DO 1 J = 1,33
    1 IX(I,J) = 1R
      CALL SECOND (KLOK)
      REWIND 1
      REWIND 3
C
C     START WHILE LOOP
C
    2 IF (EOF1.NE.0.) GOTO 900
      VERT = 0
      HOR = 0
C
C     LEES OPLOSSING
C
    3 READ(1,96) ((IX(VERT+J+1,HOR+17-I), J = 1, 4), I = 1, 15)
      EOF1 = EOF(1)
      IF (EOF1 .NE. 0.) GO TO 40
   96 FORMAT(1X,60R1)
      DO 4 I = 1,15
      DO 4 J = 1,4
      IF(IX(VERT+J+1,HOR+17-I) .EQ. 1R ) GOTO 40
    4 CONTINUE
C
C     END LEESOPLOSSING
C
      AANTAL = AANTAL+1
      IF (HOR) 5,6,5
    6 HOR = 16
      GOTO 7
    5 HOR = 0
      VERT = VERT + 5
    7 IF ( VERT .LE.20 .AND. EOF1.EQ.0. ) GOTO 3
C
C     END VAN REPEAT
C
   40 N = AANTAL - LAST
      IF (N .EQ. 0) GO TO 900
      IF (N/2 * 2 .EQ. N) GOTO 9
C
      VERT = (N-1)/2 * 5
      HOR = 16
      DO 8 I = 1,15
      DO 8 N = 1,4
    8 IX(VERT + N + 1, HOR + I + 1) = 1R
C
C     PRINT OPLOSSING
C
    9 PAG = PAG + 1
      PCC = 1R0
      LAST1 = LAST + 1
      WRITE(3, 99) LAST1, AANTAL, PAG
   99 FORMAT(1OH1 NUMMERS ,I4,6H  TOT ,I4,60X,5HBLZ. ,I4)
      MAX = (AANTAL - LAST + 1)/2 * 5
C
      DO 30 I = 1,MAX
       DO 18 J = 1,32
      A = IX(I,J)
      B = IX(I,J+1)
      C = IX(I+1,J)
       D = IX(I+1,J+1)
      IF(A-C) 10,11,10
   11  IF (B-D) 12,13,12
   13   IF (A-B) 14,15,14
```

```
   15    UU(J) = STR(1)
         GOTO 18
   14 UU(J) = STR(2)
         GOTO 18
   12 UU(J) = STR(3)
         GOTO 18
   10 IF((A .EQ. B) .AND. (C .EQ. D)) GOTO 16
         UU(J) = STR(4)
         GOTO 18
   16 UU(J) = STR(5)
   18 CONTINUE
         WRITE(3,97) PCC, (UU(J),J = 1,32)
   97 FORMAT (R1,10X,2(16R3,10X))
C
C        END EERSTE REGEL
C
         DO 24 J = 1,32
         IF (IX(I+1,J) .EQ. IX(I+1,J+1)) GOTO 20
         UU(J) = STR(2)
         GOTO 24
   20 UU(J) = STR(1)
   24 CONTINUE
   26 WRITE(3,97) 1R , (UU(J),J = 1,32)
C
C        END TWEEDE REGEL
C
         IF (I/5 * 5 .EQ. I) GOTO 27
         PCC = 1R
         GOTO 30
   27 PCC = 1R0
C
   30 CONTINUE
C
         LAST = AANTAL
C
C        END PRINT OPLOSSING
C
         GOTO 2
C
  900 CALL SECOND(HULP)
         HULP = HULP * KLOK
  909 FORMAT(16H1EXECUTIETIJD : ,
        1 1F11.3, 5H SEC.,/,23H AANTAL OPLOSSINGEN IS , I5)
         WRITE(2,909) HULP, AANTAL
         STOP
         END
```

```
(*$T+ *)
PROGRAM LISTPENTOPL(INPUT, UITV, OUTPUT);
   (* JKOK, 750604, PRINT OPLOSSINGEN VAN 15 * 4 PENTOMINO PROBLEEM *)

  LABEL (* EXIT *) 1;

  VAR I, N, AANTAL, PAG, LAST, HOR, VERT, KLOK : INTEGER;
      X : ARRAY [0 .. 25, 0 .. 32] OF CHAR;
      UITV : TEXT;

  PROCEDURE PRINTOPL;
     VAR I, J, MAX : INTEGER;
         A, B, C, D, PCC : CHAR;
  BEGIN PAG:= PAG + 1; PCC:= '0';
     WRITELN(UITV, '1 NUMMERS ', LAST + 1 : 4, '  TOT ', AANTAL : 4,
     ' ' : 60, 'BLZ. ', PAG : 4);
     MAX:= (AANTAL - LAST + 1) DIV 2 * 5 - 1;
     FOR I:= 0 TO MAX DO
     BEGIN WRITE(UITV, PCC, ' ' : 10);
        FOR J:= 0 TO 31 DO
        BEGIN A:= X[I,J]; B:= X[I,J + 1]; C:= X[I + 1, J];
           D:= X[I + 1, J + 1];
           IF A = C THEN
           BEGIN IF B = D THEN
              BEGIN IF A = B THEN WRITE(UITV, '   ') ELSE
                 WRITE(UITV, '  |')
              END ELSE WRITE(UITV, '  +')
           END ELSE
           IF (A = B) AND (C = D) THEN WRITE(UITV, '---') ELSE
           WRITE(UITV, '--+');
           IF J = 15 THEN WRITE(UITV, ' ' : 10)
        END;
        WRITELN(UITV); WRITE(UITV, ' ' : 11);
        FOR J:= 0 TO 31 DO
        BEGIN IF X[I + 1, J] = X[I + 1, J + 1]
           THEN WRITE(UITV, '   ') ELSE WRITE(UITV, '  |');
           IF J = 15 THEN WRITE(UITV, ' ' : 10)
        END;
        WRITELN(UITV); IF I MOD 5 = 4 THEN PCC:= '0' ELSE PCC:= ' '
     END (* FOR I *);
     LAST:= AANTAL
  END (* PRINT OPL *);

  PROCEDURE LEESOPL;
     VAR I, J : INTEGER; H : CHAR;
  BEGIN READ(H);
     FOR I:= 1 TO 15 DO
     FOR J:= 1 TO 4 DO
     BEGIN READ(H); X[VERT + J, HOR + 16 - I]:= H;
       IF H = ' ' THEN GOTO (* EXIT *) 1
     END; READLN
  END (* LEES OPL *);

BEGIN (* LIST PENT OPL *)
  AANTAL:= 0; PAG:= 0; LAST:= 0; FOR I:= 0 TO 25 DO
  UNPACK('                                ', X[I], 0);
  KLOK:= CLOCK; RESET(INPUT); REWRITE(UITV); LINELIMIT(UITV, 3000);
  WHILE NOT EOF(INPUT) DO
  BEGIN VERT:= 0; HOR:= 0;
     REPEAT LEESOPL; AANTAL:= AANTAL + 1; IF HOR = 0 THEN HOR:= 16
        ELSE BEGIN HOR:= 0; VERT:= VERT + 5 END
     UNTIL (VERT > 20) OR EOF(INPUT);
     N:= AANTAL - LAST; IF N MOD 2 <> 0 THEN
     BEGIN VERT:= (N - 1) DIV 2 * 5; HOR:= 16;
        FOR I:= 1 TO 15 DO
        FOR N:= 1 TO 4 DO X[VERT + N, HOR + I]:= ' '
     END;
     PRINTOPL
  END;
(* ERROR  EXIT *) 1 :
  WRITELN('0EXECUTIETIJD : ', (CLOCK - KLOK) / 1000 : 11: 3, ' SEC.');
  WRITELN(' AANTAL OPLOSSINGEN IS ', AANTAL : 5)
END .
```

ONTVANGEN 3 JUNI 1976