Zoekprocessen in het menselijk geheugen : een model en twee experimenten

van den Eijnde, P.F.H.M.

Award date:
1984

ZOEKPROCESEN IN HET MENSELIJK GEHEUGEN:
een model en twee experimenten

Afstudeerverslag van P.F.H.M. van den Eijnde
met begeleiding van L.J. Thijs

DEEL 2: BIJLAGEN

Vakgroep PPAD
Onderafdeling W&Mw
Technische Hogeschool Eindhoven
oktober 1984

Inhoudsopgave:

§B1: Het besturingsprogramma

§B1.1: Inleiding

In §4 is beschreven aan welke voorwaardes het door ons gemaakte besturingsprogramma moest voldoen, en welke mogelijkheden het heeft. In deze bijlage wordt de opbouw en de werking ervan besproken.

Uit praktische overwegingen is het besturingsprogramma opgebouwd uit een aantal zelfstandig functionerende programmas. In volgorde, waarin ze gebruikt dienen te worden zijn dat:

- SCRAMB
- INTAKT
- SEQWOR
- PSYCHO

Tussen deze 4 hoofdprogrammas, wordt informatie uitgewisseld via files (zie fig.B1.1). Bij de omschrijving van de afzonderlijke programmas zal ook de inhoud van deze files aan de orde komen.

In de volgend § worden de 4 programmas achtereenvolgens beschreven. Van SCRAMBLE en PSYCHO zullen slechts de input en de output ter sprake komen. Dit heeft twee redenen:

- De 2 programmas zijn niet door mij gemaakt.
- Naar alle waarschijnlijkheid behoeven ze binnen het kader van het geplande onderzoek (zie §1) geen wijziging te ondergaan.

Voorafgaand aan de in §B1.4 en par B1.5 beschreven programmas INTAKT en SEQWOR , worden in §B1.3 een aantal variabelen en arrays gedefinieerd, die in deze 2 programmas dezelfde betekenis hebben. Deze variabelen en arrays bevatten de waardes van de experimentele parameters.

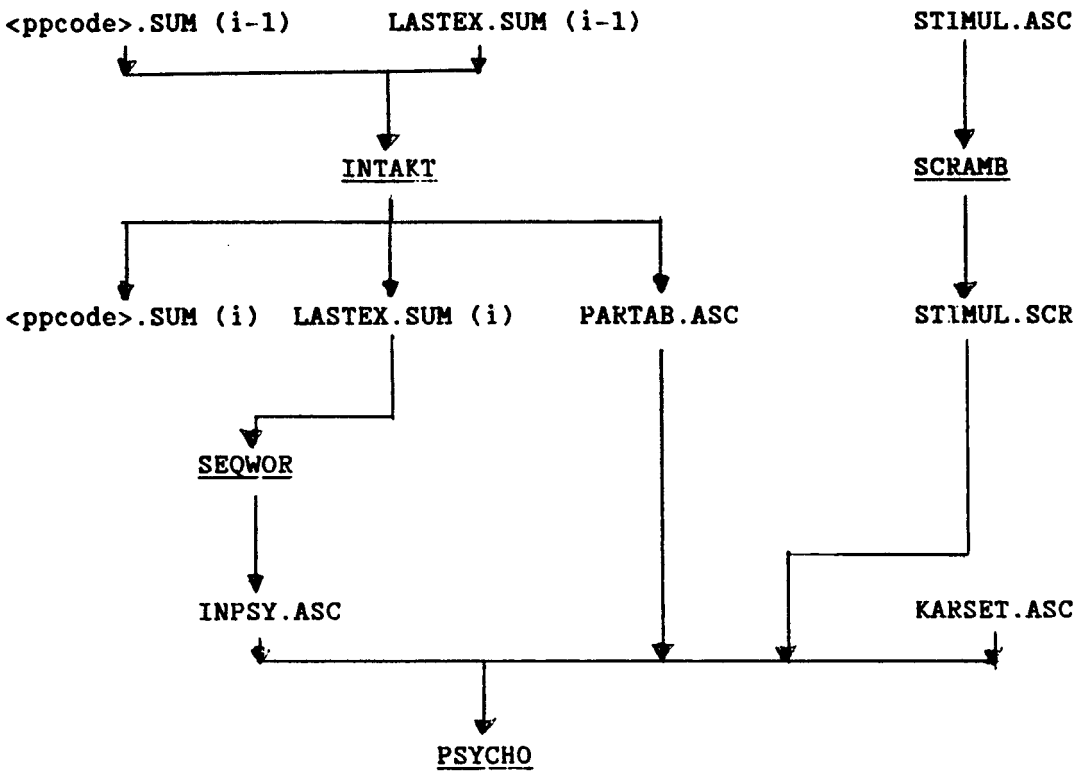

fig. 1.1.1: Tussen de 4 hoofdprogrammas, (de in dit schema onderstreepte woorden) wordt informatie uitgewisseld via files (de in dit schema niet onderstreepte woorden)

NB: gaat er een pijl van de file naar het programma, dan is het een inputfile voor dit programma; in het omgekeerde geval een outputfile

NB: tussen haakjes bij de files staan de sessienummers

§B1.2: SCRAMBLE

Inputfiles:

<filenaam>.ASC: Dit is een sequential data-file met een willekeurige naam en variabel aantal regels "NR", die door het programma worden opgevraagd. Het file dient per regel een woord (i.h.a.: een ASCI-string, zonder spaties) te bevatten. In het in §5 beschreven experiment werd hiervoor het file SCRAMBLE.ASC gebruikt, waarin op alfabetische volgorde de ruim 3400 zelfstandige naamwoorden (zie §4.1; Stimuli).

Outputfiles:

STIMUL.SCR: Dit is een data-file, dat alle woorden uit <filenaam>.ASC bevat, maar dan in pseudorandom volgorde en in blokken van 48. Het dient als input-file voor PSYCHO.
NB: Omdat het uitlezen van dit file door PSYCHO sterk tijdkritisch is, wordt in het file zelf een ASSEMBLER-procedure opgenomen. Daarom kan dit file niet door een FORTRAN programma geopend worden.

Functieomschrijving:

- De naam (<filenaam>.ASC) en de lengte van een data-file, dat al aanwezig dient te zijn, worden opgevraagd.
- De woorden uit het file <filenaam>.ASC worden in pseudorandom volgorde in blokken van 48 in het door SCRAMBLE geopende file SCRAMBLE.SCR gezet.

  Aan SCRAMBLE.SCR wordt een ASSEMBLER-procedure gelinkt ter vergroting van de snelheid, waarmee het uitgelezen kan worden.

§B1.3: De experimentele parameters

In deze § worden een aantal variabelen en arrays gedefinieerd, die in de programmas INTAKT en SEQWOR (zie §B1.4 resp. §B1.5) dezelfde betekenis hebben. Elke hier gedefinieerde variabele en array, waarvan de naam niet met een "I" begint (het volgende geldt ook voor INSTRL en INSTRT, hoewel die met een "I" beginnen.), correspondeert met (een aantal) experimentel(e) parameter(s). In INTAKT krijgt zo'n variabele (of array) een waarde, die in SEQWOR verder constant blijft.
Hieronder volgen de definities:

NBLOKL, NBLOKT: (integer) het totale aantal aanbiedingsblokken, waaruit de leerfase resp. testfase is opgebouwd.
NB: 1 <= NBLOKL, NBLOKT <= 48

NDIFL,NDIFT: (integer) het aantal verschillende aanbiedingsblokken, waaruit de leerfase resp. testfase is opgebouwd.
NB: 1 <= NDIFL, NDIFT <= 4

IDIFL,IDIFT: (integer) een nummer corresponderend met een type aanbiedingblok in de leerfase resp. testfase
NB: 1 <= IDIFL,IDIFT <= NDIFL, NDIFT

NIDEML,NIDEMT: (integer) het aantal cycli: een cyclus is een sequentiele opvolging van de verschillende aanbiedingsblokken (elk type blok éénmaal), in de leerfase resp. testfase .
NB: NIDEML=NBLOKL/NDIFL; NIDEMT=NBLOKT/NDIFT.

IIDEML,IIDEMT: (integer) een nummer corresponderend met een cyclus in de leerfase resp. testfase
NB: 1 <= IDIFL,IDIFT <= NDIFL, NDIFT

NABL,NABT: (integer array (1:4)) het totale aantal aanbiedingen per blok in de leerfase resp. testfase.
NB: 1 <= NABL(IDIFL), NABT(IDIFT) <= 999

TAL,TAT: (integer array (1:4)) de aanbiedingstijden (in secondes/10) per blok in de leerfase resp. testfase.
NB: 1 <= TAL(IDIFL), TAT(IDIFT) <= 90
NB: binnen een blok zijn de aanbiedingstijden dus constant, tussen blokken kunnen deze varieren.

TTL,TTT: (integer array (1:4)) de tussentijden (tussen 2 aanbiedingen in secondes/10) per blok in de leerfase resp. testfase.
NB: 1 <= TTL(IDIFL), TTT(IDIFT) <= 90
NB: binnen een blok zijn de tussentijden dus constant, tussen blokken kunnen deze varieren.

INSTRL,INSTRT: (logical array (1:4)) per blok of het wel of niet (.FALSE. resp. .TRUE.) voorafgegaan moet worden door een instructie.

NTYABL, NTYABT: (integer array (1:4)) de aantallen types aanbiedingen per blok in de leerfase resp. testfase.
NB: 1 <= NTYABL(IDIFL), NTYABT(IDIFT) <= 9.

ITYABL,ITYABT: (integer) een nummer corresponderend met een type-code in blok IDIFL, of IDIFT in de leerfase resp. testfase
NB: 1 <= ITYABL, ITYABT <= NTYABL(IDIFL), NTYABT(IDIFT)

TYPEAL, TYPEAT: (integer array (1:4, 1:9) de codes van de aanbiedingstypes per blok in de leerfase resp. testfase.
NB: 1<= TYPEAL(IDIFL,ITYABL), TYPEAT(IDIFL,ITYABL) <= 99

NPTYAL, NPTYAT: (integer array (1:4, 1:9) de aantallen aanbiedingen per aanbiedingstype per blok in de leerfase resp. testfase.

NB: NPTYAL(IDIFL,ITYABL), NPTYAT(IDIFL,ITYABL) moeten even zijn als woordtype 2 of 7 in het corresponderende aanbiedingstype voorkomen.

NB: 1<=NPTYAL(IDIFL,ITYABL), NPTYAT(IDIFL,ITYABL) <= 999

MININ, MAXIM: (integer) nummers corresponderd met het eerste resp. laatste voor de sessie gereserveerde woord in het file STIMUL.SCR

§B1.4: INTAKT

§B1.4.1: Inputfiles

<ppcode>.SUM: Dit is een file, dat o.a. de experimentele ("oude") parameterwaardes bevat, van alle voorafgaande experimenten van dezelfde proefpersoon. De proefpersooncode wordt aan de gebruiker opgevraagd. De experimentele parameters voor het nieuwe experiment krijgen als default-waardes deze "oude" parameterwaardes.
Verder is op dit file bijgehouden, hoeveel woorden van STIMUL.SCR de bewuste proefpersoon al gehad heeft.

LASTEX.SUM: Dit is een file, dat de experimentele ("oude") parameterwaardes bevat, van het laatste voorafgaande experiment, eventueel van een andere proefpersoon. Bij afwezigheid van het file: <codepp>.SUM krijgen de experimentele parameters voor het nieuwe experiment als default-waardes deze "oude" parameterwaardes.
NB: is op het data-floppy ook dit file niet aanwezig, dan worden deze default-waardes: 0.

§B1.4.2: Outputfiles

<ppcode>.SUM: zie "Inputfiles"; Aan dit file worden de (eventueel door de proefleider gewijzigde) parameterwaardes voor de nieuwe experimentele sessie toegevoegd. Het file dient als inputfile voor INTAKT, mits er na deze experimentele sessie nog een volgt.

LASTEX.SUM: zie "Inputfiles"; De (eventueel door de proefleider gewijzigde) parameterwaardes voor de nieuwe experimentele sessie worden over de oude heengschreven. Het file dient eerst als inputputfile voor SEQWOR, vervolgens voor INTAKT, mits er na deze experimentele sessie nog een volgt.

PARTAB.ASC: Dit file bevat (na afloop van INTAKT):

- een rijtje labels; elk label correspondeert met een aanbiedingsblok (zie §B1.3; SEQWOR); de volgorde van het rijtje bepaalt de volgorde van de aanbiedingsblokken.
- per verschillend label drie parameters van het corresponderende aanbiedingsblok:
- aanbiedingstijd
- tijd tussen twee aanbiedingen
- een integer, die de volgende waardes kan hebben:

0: aanbiedingstijd = max. aanbiedingstijd

1: aanbieding wordt na één knopindruk van de proefpersoon afgebroken. (mits aanbiedingstijd .lt. max. aanbiedingstijd

2: aanbieding wordt na twéé knopindrukken van de proefpersoon afgebroken. (mits aanbiedingstijd .lt. max. aanbiedingstijd

§B1.4.3: Globale functieomschrijving

INTAKT is een interaktief programma, dat als functie heeft :

- De proefleider te informeren over de "oude" parameterwaardes: dat zijn de waardes van de experimentele parameters van de laatste voorafgaande sessie van de proefpersoon; of bij een eerste sessie de parameterwaarden van de laatste door een andere proefpersoon verichte sessie.
- door de proefleider ingevoerde wijzigingen te controleren op een aantal kriteria, en vervolgens door te voeren.
- de aldus verkregen nieuwe parameterwaardes naar de outputfiles weg te schrijven.

§B1.4.4: Gedetailleerde functieomschrijving

- In het BLOCKDATA-subprogramma krijgt FILNMS als defaultwaarde: DX1:000000.SUM
- De variabelen en COMMON-BLOCKS worden gedeclareerd.
- Een aantal monitor-parameters krijgen een waarde; het scherm wordt schoongeveegd (zie "CHSCRE").
- eerst wordt gezocht naar een reeds bestaand file LASTEX.SUM; is dit niet aanwezig (FIRSPP=.TRUE.) dan wordt een nieuw file met die naam geopend.
- Het scherm wordt schoongeveegd, en de cursor op de eerste positie gezet (zie CHSCRE); de proefpersooncode wordt opgevraagd en in FILNMS weggeschreven. (zie PPCODE)
- eerst wordt gezocht naar een reeds bestaand file <ppcode>.SUM; is dit niet aanwezig (FIRSES=.TRUE.) dan wordt een nieuw file met die naam geopend.
  De experimentele parameters worden in de daarvoor bestemde variabelen ingelezen (zie "Formele parameters") uit:
  - <ppcode>.SUM; als FIRSES=.FALSE.
  - LASTEX.SUM; als FIRSES=.TRUE. en FIRSPP=.FALSE.
- Is: FIRSES=.TRUE. en FIRSPP=.TRUE., dan krijgen de experimentele parameters de waarde: 0.

- Als de gebruiker nog veranderingen in de parameterwaardes aan wil brengen (CHANGE=.TRUE.) worden de parameters met bijbehorende waarden op het scherm geprojecteerd (zie "NAMGEG"); en vervolgens ingelezen (zie "LEES")
- Het voorgaande stuk wordt herhaald tot de gebruiker tevreden is (CHANGE=.FALSE.)
- Aan de hand van de aanbiedingstype-codes met bijbehorende aantallen, opgeslagen in TYPEAL resp. NPTYAL wordt bepaald hoeveel woorden voor de leerfase nodig zijn.
  NB: voor type-code 2 en 7 bedraagt het aantal benodigde woorden maar de helft van het aanbiedingsaantal. (zie §4.3)
- Aan de hand van de aanbiedingstype-codes met bijbehorende aantallen, opgeslagen in TYPEAT resp. NPTYAt wordt bepaald hoeveel woorden met type-code 9 voor de testfase nodig zijn.
  NB: Alle woorden met een andere type-code worden minimaal even vaak in de leerfase aangeboden.
- TELWOR is het totaal aantal voor deze sessie benodigde woorden; met TELWOR en MAXIM, het in eerdere sessies gebruikte aantal woorden, wordt bepaalt welk stuk van SCRAMBLE.SCR (MINIM EN MAXIM) gebruikt wordt voor deze sessie.
- De naam van het door PSYCHO voor deze sessie aan te maken data-file wordt samengesteld uit de proefpersoon-code (4 letters) en NSESSY (2 letters).
- De nieuwe parameterwaardes worden naar LASTEX.SUM en /ppcode/.SUM weggeschreven.
- Op grond van de nieuwe parameterwaardes wordt PARTAB.ASC overgeschreven.
- De diverse files worden gesloten, en de monitorparameters worden in de oude staat herstelt (zie CHSCRE: SETPAR)

§B1.4.5: Subroutines van INTAKT

In deze § worden de subroutines beschreven, welke aangeroepen worden door het programma: INTAKT . De source files van deze subroutines staan op het de achterkant van het floppie "Peter1". Het zijn:

naam subroutine:	file:
PPCODE	DX1:CODEPP.FOR
SETPAR	DX1:CHSCRE.FOR
POCURS	,, ,, ,,
ERPART	,, ,, ,,
SKLEUR	,, ,, ,,
ANSWER	,, ,, ,,
NAMGEG	DX1:NAMGEG.FOR
LEES	DX1:LEES.FOR
LELOFT	DX1:LELOFT.FOR

In onderstaand schema (zie fig.1.4.4.1) staat voor iedere procedure aangegeven welke andere procedures daardoor aangeroepen worden.

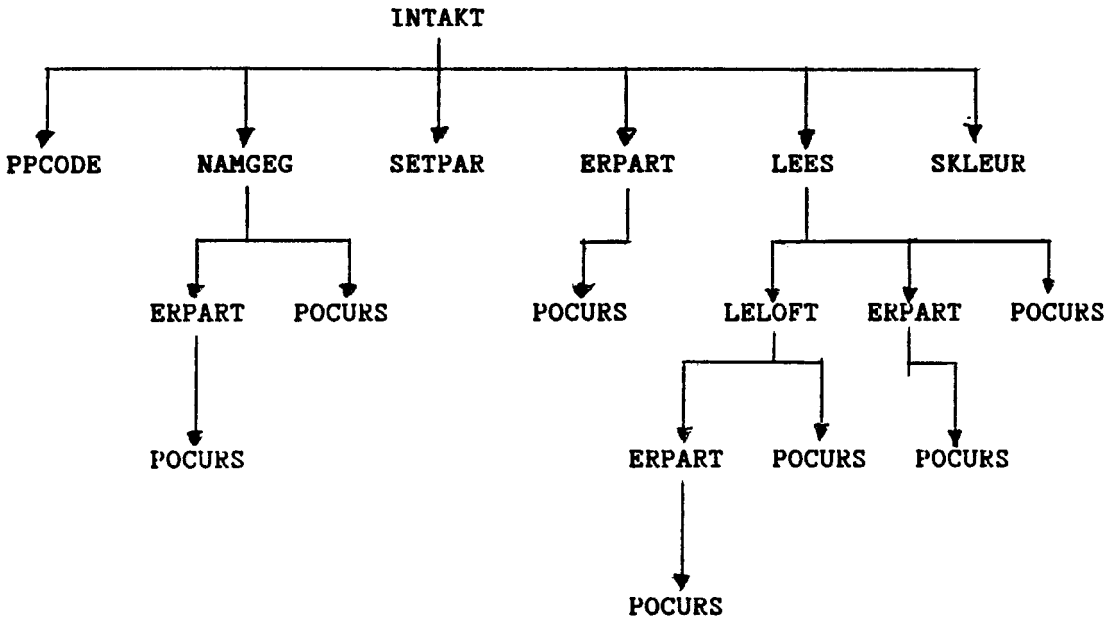

fig.: 1.4.4.1 In dit schema staat door middel van een pijl voor iedere procedure aangegeven welke andere procedures daardoor aangeroepen worden.

PPCODE:

Functieomschrijving:

- De procedure vraagt naar de code van de pp en leest deze in vanaf de terminal in het array TEXT(14).
- Hierna wordt de code op 2 kriteria getest:
  aantal karakters NK≤6 ?
  is het eerste karakter een letter ?
- Is aan deze kriteria voldaan, dan wordt gevraagd, of de code nog veranderd moet worden. Zo niet (CHANGE=.FALSE.), dan wordt de code in het array FILNMS gezet op de plaatsen 5 t/m 4+NK. Aan de overige array-elementen is in het hoofdprogramma (het BLOCKDATA-gedeelte) al een waarde toegekend. Zo ja (CHANGE=.TRUE.), dan start de procedure van voren af aan.
- Is aan bovenstaande 2 kriteria niet voldaan, dan start de procedure van voren af aan.

Formele en COMMON variabelen:

FILNMS: Op de plaatsen 5 t/m 4+NK staat na afloop van de procedure een pp-code. Aan de overige array-elementen is in het hoofdprogramma (het BLOCKDATA-gedeelte) al een waarde toegekend. Uiteindelijk staat in dit array dan een file-naam: DX1:",<ppcode>,(6-NK)x"0",".SUM.

ANSWER

Functieomschrijving:

De procedure leest een antwoord vanaf de terminal in in ANSW. Er zijn nu 3 mogelijkheden:

- ANSW="Y": dan wordt LOANSW=.TRUE.
- ANSW="N": dan wordt LOANSW=.FALSE.
- ANSW heeft een andere waarde: op de terminal verschijnt een foutmelding, en de procedure begint van voren af aan.

Formele en COMMON variabelen:

LOANSW: bevat na afloop de waarde .TRUE. of .FALSE..

SETPAR,POCURS,ERPART,SKLEUR

Voordat ik een functiebeschrijving van de in de file DX1:CHSCRE.FOR opgeslagen procedures zal geven, wil ik eerst een voor alle 4 belangrijke opmerking plaatsen:

De PDP11 is aangesloten op een Gigi-terminal. Men kan met de Outputstatements (WRITE, TYPE ,PRINT) ook niet-schrijf-opdrachten aan deze terminal doorgeven door het wegschrijven van een zogenaamde Command-string, d.i. een string, welke begint met een (ESC) (= 033 octaal). Zulke Commands kunnen bijv. zijn: het wijzigen van een SETUP-parameter, zoals: de kleur van het beeldscherm (zie SKLEUR); of het wijzigen van de cursorpositie (zie POCURS); of het schoonvegen van een aantal regels van het scherm (zie ERPART). Voor verdere informatie zie "Gigi,VK100: Installation and owner's manual".

SETPAR

Functieomschrijving:

Wil men één of meer van de andere procedures van het file DX1:CHSCRE.FOR gebruiken, dan moet men in het hoofdprogramma aan het begin SETPAR(.TRUE.) aanroepen en aan het einde SETPAR(.FALSE.). De gevolgen hiervan zijn:

- ESC krijgt de waarde "033" octaal

Als PARSET=.TRUE.:

- De scroll-mode (SM) wordt SM=3; dit heeft tot gevolg, dat:
  - Als de cursor op de onderste regel van het scherm staat, en (RETURN) wordt ingedrukt de cursor naar het begin van de bovenste regel springt.
  - Alle regels op het scherm blijven ongewijzigd tot er overheen wordt geschreven.
- het scherm wordt schoongeveegd.
- De cursor komt links boven op het scherm te staan

Als PARSET=.FALSE:

- SM=2; dit heeft tot gevolg, dat:
  - Als de cursor op de onderste regel van het scherm staat en (RETURN) wordt ingedrukt, alle regels één regel naar boven schuiven; Boven verdwijnt dan een regel, en onder komt er een bij.

    NB: SM=2 is de default-waarde.
- Cursor en bijbehorende eigenschappen als kleur worden weer in de oude toestand gebracht.
- De cursor wordt aan het begin van de 20e regel gezet.

Formele en COMMON variabelen:

PARSET: moet bij de eerst aanroep de waarde .TRUE. hebben; bij de 2e aanroep de waarde .FALSE;.

ESC: verkrijgt na aanroep SETPAR de waarde "033" octaal.

POCURS

Functieomschrijving:

Deze procedure zet de cursor op de gewenste plaats op het scherm. Deze plaats wordt bepaald door een regelnr. PL, en een kolomnr. PC.

Formele en COMMON variabelen:

ESC: heeft waarde "033" octaal; zie SETPAR

PL: moet bij aanroep procedure het gewenste regelnr. bevatten.
NB: PL.LE.23

PC: moet bij aanroep procedure het gewenste kolomnr. bevatten.
NB: PL.LE.80

ERPART

Functieomschrijving:

Veegt een rechthoekig deel van het scherm schoon; nl. de rechthoek met hoekpunten: (PL,PC), (PL,80) (PL+NLINE-1,PC), (PL+NLINE-1,80). Dit geschiedt door NLINE keer de volgende stappen te herhalen:

- De cursor wordt met behulp van de procedure POCURS gezet op de positie PL+I-1,PC.
- De regel wordt vanaf deze positie tot aan het eind schoongeveegd.
- I=I+1

Formele en COMMON variabelen:

PL: regelnr. van bovenste regel, waarvan een gedeelte moet worden schoongeveegd; $0 \leq PL \leq 23$.

PC: kolomnr. van de meest linkse kolom, waarvan een gedeelte moet worden uitgeveegd.

NLINE: aantal regels, waarvan een gedeelte moet worden schoongeveegd.

ESC: heeft waarde octaal "033"; zie SETPAR.

SKLEUR

Functieomschrijving:

Deze procedure wijzigt de kleur, waarin de output op het scherm wordt geschreven. Er ziin 8 mogelijke kleuren.

Formele en COMMON variabelen:

ESC: heeft de waarde "033" octaal (zie SETPAR).

KLEUR: afhankelijk van de waarde van KLEUR wordt de schrijfkleur bepaalt:

KLEUR:	schrijfkleur:
30	zwart
31	rood
32	groen
33	geel
34	blauw
35	magenta
36	cyaan
37	wit

NAMGEG:

functieomschrijving:

We kunnen twee gevallen onderscheiden:

- FIRSQ=.TRUE.: Op de eerste 3 regels van het beeldscherm wordt in geel de omschrijving gegeven van een aantal experimentele parameters (zie §B1.3), en in groen de oude waardes ervan.

- FIRSQ=.FALSE.: Op de regels 4 t/m 22 van het beeldscherm wordt in geel de omschrijving gegeven van de overige experimentele parameters (zie §B1.3), en in groen de oude waardes ervan.

Formele en COMMON variabelen:

Alle met de experimentele parameters corresponderende variabelen en arrays (zie §B1.3.)

FIRSQ: (logical) a.h.v FIRSQ wordt bepaald of de eerste 3 of de overige regels beschreven moeten worden.

LEES:

functieomschrijving:

We kunnen twee gevallen onderscheiden:

REGL23=.TRUE.:

Op regel 2 en 3 zijn door NAMGEG de oude waardes van een aantal parameters gegeven. Door LEES worden voor beide regels de volgende stappen doorlopen:

- de foutmeldings-regel (regel 23) wordt schoongeveegd (zie ERPART); PARNEW wordt op 0 geinitialiseerd; en de cursor wordt op de bewuste regel op positie 30 gezet (zie POCURS).
- de door de gebruiker ingetypte gegevens worden ingelezen.
  NB: is alleen een <RETURN> ingetypt, dan blijven de met de regel corresponderende gegevens ongewijzigd (label 100).
- de ingelezen gegevens worden getest op een aantal kriteria (zie §B1.3). Er zijn nu twee mogelijkheden:
  - de ingelezen gegevens voldoen aan de gestelde criteria:
    - er wordt gesprongen naar label 90
    - de ingelezen gegevens worden in de met de regel (IREGEL=2, of 3) corresponderende variabelen ingelezen.
    - er wordt naar label 90 gesprongen (einde subroutine)
  - de ingelezen gegevens voldoen niet aan de gestelde criteria:
    - er wordt afhankelijk van het incorrecte ingelezen gegeven gesprongen naar label 1100 t/m 1200
    - op regel 23 wordt in rood een foutmelding gegeven
    - er wordt naar label 5 gesprongen, zodat alle stappen van vooraf aan doorlopen worden.

REGL23=.FALSE.:

De overige experimentele parameterwaardes worden ingelezen, gecontroleerd, en toegekend aan de daarmee corresponderende variabelen en arrays. Dit gebeurt door eerst voor de leerfase en vervolgens voor de testfase de subroutine LELOFT aan te roepen.

Formele en COMMON variabelen:

Alle met de experimentele parameters corresponderende variabelen en arrays (zie §B1.3.)

REGL23: (logical) a.h.v REGL23 wordt bepaald of de met regel 2 of 3 corresponderende gegevens of de overige gegevens ingelezen moeten worden.

LELOFT:

functieomschrijving:

- de cursor wordt op regel PL0 gezet.
- er wordt NDIF maal (corresponderende met het aantal aanbiedingsblokken in de beschouwde fase) de volgende stappen doorlopen.
  - de foutmeldings-regel (regel 23) wordt schoongeveegd.
  - de cursor wordt 2 regels lager gezet op de eerste positie.
  - PARNEW wordt op 0 geinitialiseerd.
  - de door de gebruiker ingetypte gegevens worden ingelezen in LINST, DKOM (dit is een dummyvariabele, die een komma opvangt.), en PARNEW.
    NB: wordt alleen een <RETURN> ingetypt (INT=0) dan wordt naar label 2500 gesprongen; de met deze regel corresponderende parameterwaardes blijven ongewijzigd.
  - de ingelezen parameterwaardes worden op een aantal kriteria getest (zie §B1.3); er zijn nu twee mogelijkheden:
    - de ingelezen gegevens voldoen aan de gestelde criteria:
    - er wordt gesprongen naar label 2120
    - de ingelezen gegevens worden in de met de regel corresponderende variabelen ingelezen.
    - er wordt naar label 2500 gesprongen (einde subroutine)
  - de ingelezen gegevens voldoen niet aan de gestelde criteria:
    - er wordt afhankelijk van het incorrecte ingelezen gegeven gesprongen naar label 3100 t/m 3400
    - op regel 23 wordt in rood een foutmelding gegeven
    - er wordt naar label 2015 gesprongen, zodat alle stappen van vooraf aan doorlopen worden.

§B1.5: SEQWOR

§B1.5.1: Inputfiles

LASTEX.SUM: Dit is een file, dat de experimentele parameterwaardes bevat, van het laatste voorafgaande experiment, eventueel gewijzigd door INTAKT (zie §B1.4)

§B1.5.2: Outputfiles

INPSY.ASC: Dit file fungeert als inputfile voor PSYCHO. Hierin worden door SEQWOR in de gewenste aanbiedingsvolgorde per woord 3 gegevens gezet:

- type-code
- nummer, corresponderend met een positie in het file STIMUL.SCR
- stopcode: de waarde van deze stopcode wordt door PSYCHO als volgt geinterpreteerd: (stopcode =)

  1: woord is eerste woord van een aanbiedingspaar

  <= 0: woord is laatste woord van een aanbiedingspaar, of een los woord.

  <= -1: woord is laatste woord van een aanbiedingsblok; de overgang naar het volgende aanbiedingsblok verloopt zonder onderbreking.

  <= -2: woord is laatste woord van een aanbiedingsblok; na aanbieding van dit woord wordt op het beeldscherm een instructie geprojecteerd. Pas na indruk van een (RETURN) start het volgende aanbiedingsblok.
  NB: woord kan tevens laatste woord van leerfase zijn.

  <= -3: woord is laatste woord van de sessie.

§B1.5.3: functieomschrijving

in de werking van SEQWOR kan men de volgende stappen onderscheiden:
- declaratie variabelen en arrays
- declaratie COMMON-blocks
- openen van de input- en output-files.
- inlezen van de nieuwe experimentele parameterwaardes uit het file LASTEX.SUM.
- wegschrijven van een code (CODEPP) samengesteld uit de code van de proefpersoon (4 letters) en het nummer van de experimentele sessie naar INPSY.ASC.
  NB: deze code fungeert later als file-naam voor het door PSYCHO aan te maken data-file.
- aanroep TOEWYS: hierdoor wordt voor elk in deze sessie voorkomende woordtype de benodigde hoeveelheid (nog ongebruikte) woorden gereserveerd in het file SCRAMBLE.SCR. De voor deze reservering belangrijke gegevens worden opgeslagen in MIMATY en TYPEW.

leerfase:

- eerste aanroep SEQAAN: Hierdoor worden voor de leerfase per aanbiedingsblok de vereiste aantallen met de aanbiedingstypes corresponderende codes in pseudorandom volgorde in het VOLGOR gezet.
  NB: In VOLGOR staat dus na deze aanroep van SEQAAN in de juiste volgorde voor alle NIDMBL cycli van NDIFL verschillende aanbiedingsblokken zo'n rijtje van aanbiedingscodes.
- aanroep schryf: Uitgaande van de experimentele parameterwaardes, MIMATY, JUMP, TYPEW, en VOLGOR wordt voor de leerfase in INPSY.ASC per (aan de proefpersoon) aan te bieden woord 3 gegevens weggeschreven (zie INPSY.ASC). De volgorde, waarin de met de gegevens corresponderende woorden in dit files staan is bepalend voor hun aanbiedingsvolgorde.

testfase:

- aanroep JUMPTE: per woordtype wordt voor 2 elkaar in de testfase direct opvolgende woorden van dit woordtype bepaald op welke afstand ze staan in het file STIMUL.ASC. Deze afstanden worden in JUMP bewaard.
- NB: in de leerfase is deze woordafstand voor alle woordtypes 1; de woordafstand in de testfase wordt zodanig genomen, dat de correlatie tussen aanbiedingsvolgorde van woorden van hetzelfde type in leerfase en testfase minimaal is.
- tweede aanroep SEQAAN: Hierdoor worden voor de testfase per aanbiedingsblok de vereiste aantallen met de aanbiedingstypes corresponderende codes in pseudorandom volgorde in het VOLGOR gezet, achter de aanbiedingscodes voor de leerfase.
  NB: In VOLGOR staat dus na deze aanroep van SEQAAN in de juiste volgorde voor alle NIDMBT cycli van NDIFT verschillende aanbiedingsblokken zo'n rijtje van aanbiedingscodes.
- aanroep schryf: Uitgaande van de experimentele parameterwaardes, MIMATY, JUMP, TYPEW, en VOLGOR wordt voor de testfase in INPSY.ASC per (aan de proefpersoon) aan te bieden woord 3 gegevens weggeschreven (zie INPSY.ASC). De volgorde, waarin de met de gegevens corresponderende woorden in dit files staan is bepalend voor hun aanbiedingsvolgorde.

§B1.5.4:Subroutines SEQWOR

In deze § worden de subroutines beschreven, welke aangeroepen worden door het programma: SEQWOR . De source files van deze subroutines staan op het de achterkant van het floppie "Peter1". Het zijn:

naam subroutine:	file:
TOEWYS	DX1:TOEWYS.FOR
RANDOM	DX1:SBSEQW.FOR
ZET	,, ,, ,,
BREEK	,, ,, ,,
SEQAAN	DX1:SEQAAN.FOR
SCHRYF	DX1:SCHRYF.FOR
JUMPTE	DX1:JUMPTE.FOR

In onderstaand schema (zie fig.1.4.4.1) staat voor iedere procedure aangegeven welke andere procedures daardoor aangeroepen worden.

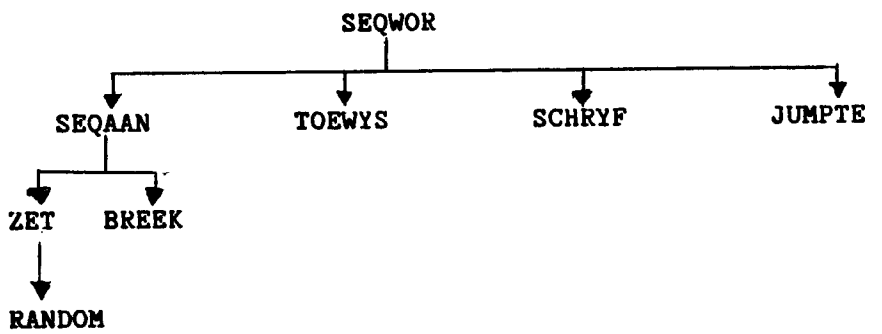

fig.: 1.4.4.1 In dit schema staat door middel van een pijl voor iedere procedure aangegeven welke andere procedures daardoor aangeroepen worden.

TOEWYS:

functieomschrijving:

- aanroep TOEWYS: hierdoor wordt voor elk in deze sessie voorkomende woordtype de benodigde hoeveelheid (nog ongebruikte) woorden gereserveerd in het file SCRAMBLE.SCR. De voor deze reservering belangrijke gegevens worden opgeslagen in MIMATY en TYPEW.

By aanroep van TOEWYS worden de volgende stappen doorlopen:

- declaraties van variabelen, arrays en COMMON-blocks.
- een aantal variabelen en arrays worden op een bepaalde waarde geinitialiseerd.

- voor de verschillend aanbiedingsblokken (IDIFL loopt van 1 tot NDIFL) wordt per aanbiedingstype het volgende vericht (ITYABL loopt van 1 tot NTYABL):
  - a.h.v. TYPEAL(IDIFL,ITYABL) wordt bekeken uit welke woordtypes het bestaat; WORD(1) en WORD(2) verkrijgen als waarde de code van het linker resp. rechter woordtype.
    NB: bestaat het aanbiedingstype uit een los woord, dan wordt WORD(1)=0.
  - a.h.v. NPTYAL(IDIFL,ITYABL) wordt het aantal benodigde woorden per woordtype (code WORD(1) resp. WORD(2)) bepaald.
  - deze codes, en bijbehorende aantallen worden opgeslagen in TYPEW en NPTYPW.
    NB: is een woordtype al in een eerder beschouwd aanbiedingstype aangetroffen, dan wordt het voor dit aanbiedingstype benodigde aantal bij het tot nu benodigde in NPTYPW opgeslagen aantal opgeteld; zo niet dan wordt er aan TYPEW een typecode toegevoegd.
    NB: voor woordtypecodes 2 en 7 hoeft maar de helft van het aanbiedingsaantal gereserveerd te worden (zie §4.3).

- de in NPTYPW opgeslagen aantallen benodigde woorden per type voor het éénmaal aanbieden van de verschillende aanbiedingsblokken worden met NIDEML (is het aantal cycli van aanbiedingsblokken in de leerfase) vermenigvuldigt teneinde de benodigde aantallen voor de gehele leerfase te verkrijgen.

testfase:

- aan de in TYPEW opgeslagen typecodes wordt de code 9 (afleider) toegevoegd. (TYPEW(ITYW) = 9)
- voor de verschillend aanbiedingsblokken (IDIFT loopt van 1 tot NDIFT) wordt per aanbiedingstype het volgende vericht (ITYABT loopt van 1 tot NTYABT):
  - a.h.v. TYPEAT(IDIFT,ITYABT) wordt bekeken uit welke woordtypes het bestaat; WORD(1) en WORD(2) verkrijgen als waarde de code van het linker resp. rechter woordtype.
    NB: bestaat het aanbiedingstype uit een los woord, dan wordt WORD(1)=0.
  - voor WORD(I) (I loopt van 1 tot 2) wordt als WORD(I) = 9 bij NPTYPW(ITYW) het aanbiedingsaantal NPTYAL(IDIFL,ITYABL) opgeteld.
    NB: alleen afleiders (code 9) kunnen vaker in de testfase voorkomen als in de leerfase.

- de in NPTYPW opgeslagen aantallen benodigde afleiders voor het éénmaal aanbieden van de verschillende aanbiedingsblokken worden met NIDEMT (is het aantal cycli van aanbiedingsblokken in de testfase) vermenigvuldigt teneinde de benodigde aantallen voor de gehele testfase te verkrijgen.

leerfase en testfase:

Voor alle woordtypes (ITYW loopt van 1 tot NTYW) worden uitgaande van MINIM, MAXIM, TYPEW en NPTYPW de benodigde gegevens in MIMATY, NEXTWL en NEXTWT opgeslagen (zie: Formele en COMMON variabelen)

Formele en COMMON variabelen:

Alle met de experimentele parameters corresponderende variabelen en arrays (zie §B1.3.)

NTYW: (integer) het totale aantal woordtypes, waaruit de aanbiedingstypes zijn opgebouwd.

TYPEW: (integer array (1:10)) de codes van deze woordtypes,

NPTYPW: (integer array (1:10)) de benodigde aantallen per woordtype

MIMATY: (integer array (1:10, 1:3)) bevat per woordtype:

- een nummer corresponderend met het eerste voor dat woordtype in SCRAMBLE.SCR gereserveerde woord
- een nummer corresponderend met het laatste voor dat woordtype in SCRAMBLE.SCR gereserveerde woord
- het benodigde aantal

NEXTWL, NEXTWT: (integer array (1:10)) bevat per woordtype: een nummer corresponderend met een voor dat woordtype in SCRAMBLE.SCR gereserveerde woord , dat als eerstvolgende aangeboden dient te worden. (in de leerfase resp. testfase)

NB: In TOEWYS wordt NEXTWL(ITYW) (ITYW loopt van 1 tot NTYW - 1) geinitialiseerd op MIMATY(ITYW,1)

NB: In TOEWYS wordt NEXTWT(ITYW) (ITYW loopt van 1 tot NTYW) geinitialiseerd op een nummer corresponderend met het middelste voor dat woordtype in SCRAMBLE.SCR gereserveerde woord:

(MIMATY(ITYW,1) + MIMATY(ITYW,2))/2

Dit om de correlatie van een woord tussen aanbiedingspositie in de leerfase resp. testfase zo klein mogelijk te houden.

RANDOM

functieomschrijving:

RANDOM geeft IRAND een pseudorandom bepaalde waarde. tussen IMIN en IMAX. Het maakt hierbij gebruik van de bibliotheekprocedures RAN, en SECONDS.

Formele en COMMON variabelen:

IRAND: krijg een pseudorandom waarde.
IMIN,IMAX: RANDOM bepaalt een pseudorandom getal: IMIN < IRA <= IMAX

ZET

Functiomschrijving:

De procedure ZET kiest pseudo-random NPTYPE arrayelementen uit van een deel (voortaan aangeduid met een blok van VOLGOR) van het array VOLGOR, dat correspondeert met een aanbiedingsblok Deze arrayelementen krijgen dan de waarde, die de variabele TYPE heeft. De pseudo-random keuze komt als volgt tot stand:

- Met de procedure RANDOM wordt aselect met teruglegging een integer IRA getrokken tussen MIINVO en MAINVO.
- Heeft VOLGOR(IRA) de waarde "0", dan wordt VOLGOR(1RA)=TYPE.
- Heeft VOLGOR(IRA) een waarde ongelijk "0", dan wordt: IRA=(IRA+11)mod(MAINVO - MIINVO). Deze stap wordt herhaald tot VOLGOR(IRA) de waarde "0" heeft. In dat geval wordt: VOLGOR(IRA)=TYPE

NB: Voorwaardes voor het functioneren van deze procedure zijn:

- voor het aantal array-elementen "NO" van het beschouwde blok van VOLGOR, dat voor de aanroep van ZET "0" is, geldt: NO≥NPTYPE
- NTOT is niet deelbaar door 11; anders komt ZET in een lus terecht.

formele en COMMON variabelen:

VOLGOR: (integer array (1:1200))het array, waarop de procedure wordt uitgevoerd.

TYPEA: (integer) bevat de waarde, die de NPTYPE pseudo-random uitgekozen arrayelementen van VOLGOR na afloop van ZET bezitten.

NPTYPA: (integer) het aantal arrayelementen van VOLGOR, dat na afloop de waarde van TYPEA heeft

MI1NVO, MAINVO: (integer) de indices van het eerste resp. laatste element van VOLGOR, dat bij het beschouwde blok behoort.

BREEK

Functieomschrijving:

In de hieronderstaande beschrijving van BREEK zal onder een "rijtje" steeds verstaan worden: een rijtje opeenvolgende arrayelementen van een blok (zie ZET) van VOLGOR met dezelfde waarde; de "waarde" van het rijtje is dan de waarde van de elementen.

BREEK test op basis van een kriterium, wat hieronder nog zal worden gegeven, of het het beschouwde blok van VOLGOR te lange rijtjes bevat. Blijkt dit zo te zijn, dan wordt door verwisseling van telkens 2 arrayelementen bewerkstelligd, dat dit niet meer het geval is. De procedure verloopt als volgt:

Uit VOLGOR wordt geselekteerd:

- het langste rijtje: MAXRY1
- het langste rijtje met een van MAXRY1 verschillende waarde: MAXRY2

Deze 2 rijtjes worden als volgt gezocht:

- We laten de index van VOLGOR, TEL, lopen van MIINVO tot MAINVO.
- Stel het programma is beland in de volgende toestand:
  - TEL=I;1<MAINVO
  - niet: VOLGOR(TEL)=VOLGOR(TEL-1)
  - de procedure gaat verder bij label 10
  - MAXRY1(1,1-3), MAXRY2(1,1-3), hebben de volgende waardes: de lengte, de index van het midden, resp. de waarde van het tot dan toe langste rijtje en het tot dan toe langste rijtje met van het eerste rijtje verschillende waarde.

de procedure gaat nu als volgt verder:

- Er wordt gecontrleerd, of TEL.GE.MAINVO; zo ja, dan is het eerste deel van de procedure afgerond, en er wordt gesprongen naar label 1000.
- zo niet, dan wordt: TELRY=0, TYPE=VOLGOR(TEL).
- TELRY en TEL worden nu net zo lang met 1 opgehoogd tot niet meer VOLGOR(TEL)=TYPE (d.i. de beginwaarde). We hebben dan een rijtje gevonden met lengte: TELRY; index van het midden: TEL-1-TELRY/2; en waarde TYPE
- Er kunnen zich nu de volgende mogelijkheden voordoen:
  - als TELRY.GT.MAXRY(1,1): (go to 600)

600:
  - als niet MAXRY(1,3)=TYPE: (go to 620)

620:
    - MAXRY(2,1-3)=MAXRY(1,1-3);
      MAXRY(1,1-3)= TELRY, TEL-1-TELRY/2, resp. TYPE (go to 10)

600:
  - als MAXRY(1,3)=TYPE: (go to 610)

610:
    - MAXRY(1,1-3)= TELRY, TEL-1-TELRY/2, resp. TYPE (go to 10)
  - als TELRY=MAXRY(1,1): (go to 500)

500:
  - als niet MAXRY(1,3)=TYPE: (go to 425)

425:
    - MAXRY(2,1-3)= TELRY, TEL-1-TELRY/2, resp. TYPE (go to 10)

500:
  - als MAXRY(1,3)=TYPE: (go to 10)
  - als TELRY<MAXRY(1,1): (go to 400)

400:
  - als TELRY>MAXRY(2,1):
    - als niet TYPE=MAXRY(1,3): (go to 425)

425:
    - MAXRY(2,1-3)= TELRY, TEL-1-TELRY/2, resp. TYPE (go to 10)
    - als TYPE=MAXRY(1,3): (go to 10)

400:
  - als TELRY<=MAXRY(2,1): (go to 10)

In alle bovenstaande gevallen komt de procedure terug bij label 10, en herhaalt zich, tenzij TEL>=MAINVO. In het laatste geval bevat MAXRY de relevante gegevens van MAXRY1 en MAXRY2, en wordt gesprongen naar label 1000.

Wanneer de procedure is aangekomen bij label 1000, kan men 2 gevallen onderscheiden:

- de lengte van MAXRY2 (MAXRY(2,1)<=3: (go to 2000)
  Dit is het aan het begin van de functieomschrijving genoemde kriterium op grond waarvan geconcludeerd wordt, dat VOLGOR geen te lange rijtjes bevat.Is aan dit kriterium voldaan, dan wordt BREEK beeindigd. (go to 2000).
- de lengte van MAXRY2 (MAXRY(2,1)>3: (go to 1300)
  De arrayelementen, met indices MAXRY(1,2), MAXRY(2,2), d.i. de middens van MAXRY1 resp. MAXRY2 worden onderling verwisseld. Hierdoor worden beide rijen opgesplitst in elk 2 deelrijen met lengte<=MAXRY(1,1)/2 resp. <=MAXRY(2,1)/2. Hierna wordt de procedure herhaald (go to 5).

Formele en COMMON variabelen:

VOLGOR: (integer array (1:1200)) In dit array worden door BREEK in het algemeen een aantal elementen onderling verwisseld ten einde minder lange rijtjes te verkrijgen.

MIINVO, MAINVO: (integer) de indices van het eerste resp. laatste element van VOLGOR, dat bij het beschouwde blok behoort.

JUMPTE

functieomschrijving:

Bepaalt voor elk woordtype,TYPEW(ITYW), een priemgetal, JUMP(ITYW), waarvan het kwadraat, PRIBFKW(ITYW) zo dicht mogelijk ligt bij het aantal voor dat woordtype gereserveerde woorden.

Formele en COMMON variabelen:

NTYW: (integer) het aantal woordtypes

TYPEW: (integer array (1:10)) bevat de codes van de woordtypes

NPTYPW: (integer array (1:10)) bevat de aantallen van de woordtypes

JUMP: (integer array (1:10)) bevat na afloop van JUMPTE de aan de gestelde criteria voldoenende priemgetallen, die corresponderen met de woordtypes.
NB: voor gebruik zie SCHRYF.

SEQAAN

functieomschrijving:

SEQAAN doorloopt (voor één fase) voor alle cycli (IIDEMB loopt van 1 to NIDEMB), over elk aanbiedingsblok (binnen zo'n cyclus; voor elke IIDEMB loopt IDIFB van 1 tot NDIFB) de volgende stappen:

- MIINVO, en MAINVO:de indices van het eerste resp. laatste element van VOLGOR, dat bij het beschouwde blok behoort, worden bepaald aan de hand van de reeds opgevulde plaatsen van VOLGOR (oude waarde van MAINVO) en het totale aantal aanbiedingen van dit blok (NAB(IDIFB))
- voor de eerste NTYAB-1 aanbiedingstypes van wordt met behulp van ZET NPTYPA(IDIFB,ITYAB) maal de code, TYPEA(IDIFB,ITYAB), op een pseudorandom bepaalde positie in het beschouwde blok gezet.
  NB: voor NTYAB=1 wordt deze stap overgeslagen.
- voor het laatste aanbiedingtype wordt in het beschouwde blok van VOLGOR overal de code, TYPEA(IDIFB,NTYAB), gezet, waar nog een 0 staat.
- met BREEK worden in het beschouwde blok van VOLGOR, eventueel aanwezige, te lange rijtjes opeenvolgende elementen met dezelfde waarde afgebroken..
  NB: voor NTYAB=1 wordt deze stap overgeslagen.

Formele en COMMON variabelen:

VOLGOR: (integer array (1:1200)) dit array bevat na afloop van SEQAAN de gewenste aantallen aanbiedingscodes in de gewenste aanbiedingsvolgorde voor de leerfase (na de eerste aanroep; zie SEQWOR) voor de leerfase én de testfase (na de tweede aanroep).

MIINVO, MAINVO: (integer) de indices van het eerste resp. laatste element van VOLGOR, dat bij het beschouwde blok behoort.

NBLOK: (integer) het totale aantal blokken

NDIFB: (integer) het aantal verschillende blokken

NIDEMB: (integer) het aantal cycli in de beschouwde fase.

NAB: (integer array (1:4)) de totale aantallen aanbiedingen per blok

NTYAB (integer array (1:4)) de aantallen types per blok

TYPEA (integer array (1:4, 1:10)) de codes van de aanbiedingstypes

NPTYA (integer array (1:4, 1:10)) de aantallen van de aanbiedingstypes

SCHRYF

functieomschrijving:

- STOPCO(1), di. de stopcode, die correspondeert met het eerste woord van een aanbiedingspaar, zodat die de (verder constante blijvend) waarde 1 krijgt.
- de stopcode wordt bepaald voor de laatst aangeboden woorden van de verschillende aanbiedingsblokken (IDIFB loopt van 1 tot NDIFB); volgt na zo'n woord een instructie (INSTRU(IDIFB+1) = .TRUE.), dan wordt deze -2, anders -1 (zie beschrijving INPSY.ASC). Deze stopcodes worden opgeslagen in STCOBU(IDIF).

- Voor alle cycli (IIDEMB loopt van 0 tot NIDEMB-1) over alle verschillende blokken (IDIFB loopt van 1 tot NDIFB) worden de volgende stappen doorlopen:
  - voor de laatste cyclus krijgt de stopcode van het laatste woord van het laatste aanbiedingsblok (STCOBU(NDIF)) de waarde -2 , of -3 voor leerfase resp. testfase (opgeslagen in STCOFA).
  - aan het begin van elk blok wordt de stopcode voor het tweede woord van een aanbiedingspaar 0 gesteld.
    NB: dit blijft deze tot aan het laatste aanbiedingspaar van het beschouwde blok.
  - voor alle aanbiedingen in het beschouwde blok (IAB loopt van 1 tot NAB(IDIF)) worden de volgende stappen doorlopen:
    - aanbiedingscode wordt gesplits in 2 woordcodes: WORD(1) en WORD(2) corresponderend met 2 woorden (is WORD(1) = 0, dan bestaat de aanbieding uit een los woord)
    - is de aanbieding de laatste aanbieding van het blok, dan wordt de stopcode van het tweede woord (STOPCO(2)) gelijk aan STOCOBU(IDIF) (zie boven)
    - voor beide woorden van het aanbiedingspaar worden de volgende stappen doorlopen (I loopt van 1 tot 2)
      - als WORD(I) de waarde 0 heeft wordt gesprongen naar label 591; het fictieve woord wordt genegeerd.
      - het nummer ITYW van het woordtype, waarmee WORD(I) correspondeert wordt bepaald.
      - de 3 voor PSYCHO relevante gegevens per woord worden weggeschreven naar INPSY.ASC.
        NB: in NEXTW(ITYW) zit het nummer waarmee een woord uit het file SCRAMB. SCR correspondeert dat gereserveerd is voor het woordtype met code WORD(I), en aan de beurt is om aangeboden te worden.
      - met behulp van MIMATY, en JUMP wordt het nummer voor het volgende aan te bieden woord van hetzelfde type (WORD(I)) bepaald, en in NEXTW(ITYW) opgeslagen.
        NB: voor de leerfase is dat eenvoudig het volgende woord uit het file SCRAMB. SCR (JUMP(ITYW) = 1; zie DATA-statement in SEQWOR); voor de testfase is dat zeker niet het geval (zie JUMJPTE).

Formele en COMMON variabelen:

VOLGOR: (integer array (1:1200)) dit array bevat bij aanvang van SCHRYF de gewenste aantallen aanbiedingscodes in de gewenste aanbiedingsvolgorde voor de leerfase (na de eerste aanroep; zie SEQWOR) voor de leerfase én de testfase (na de tweede aanroep).

NIDEMB: (integer) het aantal cycli in de beschouwde fase.

NDIFB: (integer) het aantal verschillende blokken

NAB: (integer array (1:4)) de totale aantallen aanbiedingen per blok

STCOFA: (integer) de waarde van de stopcode van het allerlaatste aan te bieden woord in deze fase.

INSTRU: (logical array (1:4)) bevat per verschillend blok de informatie of het wel of niet voorafgegaan wordt door een instructie.

NEXTW: (integer array (1:10) bevat voor elk woordtype het nummer, corresponderend met een woord in het file STIMUL.ASC, van het eerstvolgende aan te bieden woord (zie ook TOEWYS).

JUMP: (integer array (1:10) bevat voor elk woordtype de afstand in het file STIMUL.ASC, tussen twee na elkaar aan te bieden woorden van dat woordtype.

MIMATY: (integer array (1:10, 1:3)) bevat per woordtype:

- een nummer corresponderend met het eerste voor dat woordtype in SCRAMBLE.SCR gereserveerde woord
- een nummer corresponderend met het laatste voor dat woordtype in SCRAMBLE.SCR gereserveerde woord
- het benodigde aantal

§B1.6:PSYCHO

§B1.6.1:Inputfiles

PARTAB.ASC: Voor beschrijving zie: §B1.4.2 (outputfiles van INTAKT)

KARSET.ASC: Deze file bevat het formaat van de letters, waarin de woorden worden aangeboden.
NB: dit formaat is overgenomen van het IPO, dat een onderzoek vericht heeft naar de invloed van de afstand van de proefpersoon tot het beeldscherm, en de grootte van het formaat van een letter op de snelheid, waarmee de proefpersoon vermoeid wordt.

INPSY.ASC: Voor beschrijving zie: §B1.5.2 (outputfiles voor SEQWOR)

§B1.6.2: Outputfiles

<ppsecode>.DAT: Deze file bevat per aanbieding van de testfase (in aanbiedingsvolgorde)
- een aanbiedingsnummer,
- een aanbiedingscode (dezelfde die de aanbieding door SEQWOR toegewezen is)
- de antwoordtijd van de proefpersoon in miliseconden
- een nummer, corresponderen met de door de proefpersoon gedane knopindruk

<ppsecode>.ASC: Deze file bevat de in beide fases aangeboden woorden in aanbiedingsvolgorde

functieomschrijving:

§B2: gebruiksaanwijzing besturingsprogramma

Er is een speciaal floppy geprepareerd voor het draaien van herkenningsexperimenten met het besturingsprogramma (zie hoofdstuk 4). Het is een systeemfloppy, waar naast de systeem-programmas, de volgende files op staan:

STIMUL.SCR
KARSET.ASC
STAREX.COM
INTAKT.SAV
SEQWOR.SAV
PSYCHO.SAV

Voor een beschrijvig van deze files, behalve STAREX.COM, zie §B1. STAREX.COM is een command-file, waarmee het experiment wordt opgestart. Door het commando " SY:STAREX" wordt achtereenvolgens INTAKT, SEQWOR, en PSYCHO gedraaid (zie onder).

Voor aanvang van het eerste experiment van een proefpersoon, dient op het geinitialiseerde data-floppy het file INSTRU.ASC te staan. In deze met de EDITOR aan te maken file moeten instructies voor de proefpersoon staan, steeds gescheiden door één " ". Er moeten minstens zoveel instructies staan als er blokwisselingen zijn, die een instructie vooraf behoeven (dit moet de gebruiker van te voren opgeven, zie onder), echter nooit minder dan 2.

Opstarten experiment:

INTAKT is een interaktief programma, dat alle benodigde gegevens opvraagt. Ik zal hieronder een voorbeeld geven van een mogelijke sessie (geplande herkennnigsexperiment 3; zie §4.3). Tussendoor zal ik steeds, voor zover nodig, de beperkingen bij het invoeren van de gegevens vermelden. Het commentaar zal ik in kleine letters schrijven, voorafgegaan door een "!", de vragen van de computer in onderstreepte hoofdletters, voorafgegaan door "COMP:", de antwoorden en commando's van de gebruiker ook in hoofdletters voorafgegaan door "GEBR:".
NB: Staat <ppcode>.SUM of LASTEX.SUM al op het data-floppy dan geeft INTAKT vanaf de derde vraag na iedere vraag zelf het antwoord. Is de gebruiker het met dit antwoord volkomens eens, dan typt hij alleen <RETURN> in, anders de nieuwe gegevens. Terwille van het overzicht laat ik deze door INTAKT gegeven antwoorden hier weg.

GEBR: SY:STAREX ! opstarten experiment ("SY" = systeem)
COMP: CODE PROEFPERSOON ? (4KARAKTERS, EERSTE KARAKTER IS LETTER)
GEBR: BETH ! naam is bijv "Bert Thijs"
COMP: WILT U DEZE NAAM NOG VERANDEREN (Y/N)
GEBR: N ! bij "Y" komt vraag naar code pp. opnieuw.
COMP: ALLE PARAMETERS ZIJN "0" GESTELD; U DIENT ZE DUS ZELF EEN WAARDE TOE TE KENNEN; KLAAR ? (Y/N)
GEBR: Y ! is een van de files LASTEX.SUM of <ppcode>.SUM op het datafloppy aanwezig, dan wordt de gebruiker gevraagd of hij nog iets in de gegevens wil veranderen.

COMP: TOTAAL AANTAL BLOKKEN, AANTAL VERSCHILLENDE BLOKKEN, AANTAL TYPES PER BLOK?

COMP: LEERFASE: GEBR: 8,2,2,2

COMP: TESTFASE: GEBR: 12,3,3,3,4

!Gevolgen (voorbeeldgeval):

leerfase: aantal cycli is 4; één cyclus bestaat uit 2 blokken en beide blokken hebben 2 aanbiedingstypen.

testfase: aantal cycli is 4; één cyclus bestaat uit 3 blokken, de eerste twee blokken hebben 3 aanbiedingstypen, de derde 4.

Algemene beperkingen voor de invoergegevens:

- totaal aantal blokken van leer- en test-fase samen < 48
- aantal verschillende blokken per fase < 4
- aantal types per blok < 9.

!Voor elk verschillend blok verschijnt nu op het scherm een vraag om verderde gegevens.

COMP: LEERFASE: INSTRUCTIE (Y/N); AANBIEDINGS- TUSSEN-TIJD; (PER BLOK:

COMP: CODE, AANTAL)

COMP: BLOK1:

GEBR: Y,15,5,11,60,34,40

COMP: BLOK1:

GEBR: Y,15,5,68,60,55,40

COMP: TESTFASE: (ZIE LEERFASE!!!!!!)

COMP: BLOK1:

GEBR: Y,90,10,11,20,55,20,99,20

COMP: BLOK2:

GEBR: Y,90,10,34,20,55,20,99,20

COMP: BLOK3:

GEBR: Y,90,10,11,20,55,20,34,20,68,20

!Gevolgen (voorbeeldgeval):

leerfase: - De leerfase is opgebouwd uit 4 cycli met 2 verschillende typen blokken (zie vorig commentaar).
- voor het eerste type blok geldt:
  - wordt voorafgegaan door een instructie
  - aanbiedings- tussentijd zijn resp 1.5 en 0.5 seconden
  - bevat 60 aanbiedingen van type 11, d.w.z. dat ze opgebouwd zijn uit twee woorden, die in de testfase niet samen als paar terugkomen.
  - bevat 60 aanbiedingen van type 68, d.w.z. dat ze opgebouwd zijn uit twee woorden, die in de testfase wel samen als paar terugkomen.
- voor het tweede type blok geldt:
  - wordt voorafgegaan door een instructie
  - aanbiedings- tussentijd zijn resp 1.5 en 0.5 seconden
  - bevat 60 aanbiedingen van type 55, d.w.z. dat ze opgebouwd zijn uit twee woorden, die in de testfase niet samen als paar terugkomen.
  - bevat 60 aanbiedingen van type 68, d.w.z. dat ze opgebouwd zijn uit twee woorden, die in de testfase wel samen als paar terugkomen.

testfase: - De testfase is opgebouwd uit 4 cycli met 3 verschillende typen blokken (zie vorig commentaar).

- voor het eerste type blok geldt:
  - wordt voorafgegaan door een instructie
  - de maximum-aanbiedingstijd en tussentijd zijn resp 9.0 en 1.0 seconden
  - bevat 20 aanbiedingen van type 11, d.w.z. dat ze opgebouwd zijn uit twee woorden, die eenmaal in de leerfase worden aangeboden, maar niet samen als paar .
  - bevat 20 aanbiedingen van type 68, d.w.z. dat ze opgebouwd zijn uit twee woorden, die eenmaal samen als paar in de leerfase worden aangeboden .
    bevat 20 aanbiedingen van type 99, d.w.z dat ze opgebouwd zijn uit twee afleiders (woorden, die niet in de leerfase zijn aangeboden).
- voor het tweede type blok geldt:
  - wordt voorafgegaan door een instructie
  - de maximum-aanbiedingstijd en tussentijd zijn resp 9.0 en 1.0 seconden
  - bevat 20 aanbiedingen van type 55, d.w.z. dat ze opgebouwd zijn uit twee woorden, die eenmaal in de leerfase worden aangeboden, maar niet samen als paar .
  - bevat 20 aanbiedingen van type 68, d.w.z. dat ze opgebouwd zijn uit twee woorden, die eenmaal samen als paar in de leerfase worden aangeboden .
    bevat 20 aanbiedingen van type 99, d.w.z dat ze opgebouwd zijn uit twee afleiders (woorden, die niet in de leerfase zijn aangeboden).

- voor het derde type blok geldt:
  - wordt voorafgegaan door een instructie
  - de maximum-aanbiedingstijd en tussentijd zijn resp 9.0 en 1.0 seconden
  - bevat 20 aanbiedingen van type 11, d.w.z. dat ze opgebouwd zijn uit twee woorden, die eenmaal in de leerfase worden aangeboden, maar niet samen als paar .
  - bevat 20 aanbiedingen van type 55, d.w.z. dat ze opgebouwd zijn uit twee woorden, die eenmaal in de leerfase worden aangeboden, maar niet samen als paar .
  - bevat 20 aanbiedingen van type 34, d.w.z. dat ze opgebouwd zijn uit twee woorden, die eenmaal samen als paar in de leerfase worden aangeboden .
  - bevat 20 aanbiedingen van type 68, d.w.z. dat ze opgebouwd zijn uit twee woorden, die eenmaal samen als paar in de leerfase worden aangeboden .

NB: verschillen tussen de types 11 en 55, resp. 34, en 68 zitten in de aard van instructie in de leerfase, voorafgaand aan de typen blokken, waartoe ze behoren (zie §4.3)

!Algemene beperkingen en aanwijzingen voor de invoergegevens:

- Bij instuctie is "Y" verschijnt v r het blok een instructie (uit INSTRU.ASC; zie boven) en start het blok na indrukken van de spatiebalk.
  Bij instuctie is "N" verloopt de overgang van het vorige naar het bewuste blok vloeiend zonder dat iets door de gebruiker gedaan hoeft te worden.
- Voor Alle gegevens per blok, behalve "instructie" dienen gehele getallen ingevoerd te worden.
- Aanbiedings- en tussen-tijd dient men op te geven in tienden van seconden (S/10). Verder geldt:
  0 < Aanbiedings- en tussen-tijd <= 90 (S/10).
- Per type in het blok dient men steeds de code, gevolgd door het aantal op te geven. het aantal opgegeven codes (en bijbehorende aantallen) moet overeenstemmen met het voor dit blok reeds eerder opgegeven "aantal types per blok"
- Voor de codes resp. het aantal per type geldt verder:
  - 1 <= type <= 99; 1 <= aantal <= 999

COMP: WILT U NOG IETS VERANDEREN? (Y/N)
GEBR: N

!Antwoordt de gebruiker in dit geval met "Y" dan start de vragenprocedure weer vanaf de vraag "aantallen blokken, aantal verschillende blokken....."
!Antwoordt de gebruiker in dit geval met "N", dan is het interaktieve gedeelte afgerond, en wordt achtereenvolgens INTAKT gestopt, SEQWOR en daarna PSYCHO gedraaid.

Voor het feitelijke starten van de leer- en test-fase, en eventueel voor de overgang tussen twee blokken dient men dan steeds de spatiebalk in te drukken.

NB: Is INTAKT klaar, dan is een file, <ppcode>00.SUM aangemaakt, danwel gewijzigd. Dit betekent, dat ook als het experiment in een later stadium wordt afgebroken, de woorden voor het bijbehorende experiment gereserveerd zijn, en bij een nieuw experiment van de proefpersoon deze woorden als gebruikt worden beschouwd, en daarmee worden overgeslagen. Wil men voor het nieuwe experiment deze woorden toch gebruiken, dan moet men het file, <ppcode>00.SUM weggooien, en INTAKT een aantal malen draaien, zodanig dat men een file <ppcode>00.SUM verkrijgt, dat identiek is aan het file, dat bestond juist voordat de mislukte sessie gestart werd, dus voor het draaien van INTAKT.

§B3: invoergegevens voor realisering geplande herkenningsexperimenten

Met het in §B2 beschreven besturings programma kunnen, overeenkomstig de doelstellingen (zie §4.1), de 4 binnen dit project geplande experimenten (zie §4.2) gedraaid worden. In deze bijlage staat per experiment (voor experiment 2,3, en 4) opgegeven, welke type-codes in de leer- en test-fase moeten worden opgenomen:

experiment 1:

leerfase: Deze fase bestaat uit één blok, waarin losse woorden éénmaal resp. twéémaal aangeboden worden, zodat de opgenomen typecodes zijn:

1, en 2

testfase: Deze fase bestaat ook uit één blok, waarin losse woorden zijn opgenomen (niet geleerd, eenmaal geleerd, tweemaal geleerd), en woordparen (beide woorden niet geleerd, eerste of tweede woord niet geleerd, beide woorden wel geleerd), zodat de opgenomen typecodes zijn:

9, 1, 2, 99, 91,19,99

experiment 3:

leerfase: Deze fase bestaat uit twee verschillende typen blokken, waarin steeds losse woorden éénmaal aangeboden worden. In het eerste bloktype zijn de aanbiedingen lang, in het tweede bloktype kort. De opgenomen typecodes zijn:

blok1: 1
blok2: 6

NB: omdat type 1 gekoppeld is aan blok 1, en type 6 gekoppeld is aan blok 2, en per bloktype een aanbiedingstijd gekozen kan worden, kunnen we het gewenste resultaat verkrijgen.

testfase: Deze fase bestaat uit één blok, waarin losse woorden zijn opgenomen (lang geleerd, kort geleerd, niet geleerd), zodat de opgenomen typecodes zijn:

1, 6, en 9

experiment 4:

leerfase: deze fase bestaat uit een aantal cycli met per cyclus 2 verschillende type blokken. In beide bloktypen worden woordparen geleerd, in het eerste bloktype onder de instructie, G, "leer elk van de twee woorden", in het tweede bloktype onder de instructie, G', "leer het woordpaar als geheel". De volgende typecodes moeten in de bloktypes opgenomen worden (mede in verband met de testfase).

blok1: 34 en 11
blok2: 67 en 55

testfase: Deze fase bestaat uit een aantal cycli met per cyclus 3 verschillende typen blokken.

In het eerste bloktype worden woordparen aangeboden, een gedeelte geleerd onder G, een gedeelte onder G', die in de leerfase niet samen als woordpaar voorkwamen, en verder een aantal uit nieuwe woorden opgebouwde paren.

In het tweede bloktype worden woordparen aangeboden, een gedeelte geleerd onder G, een gedeelte onder G', die in de leerfase wel samen als woordpaar voorkwamen, en verder een aantal uit nieuwe woorden opgebouwde paren.

In het derde bloktype worden zowel woordparen aangeboden, die in de leerfase niet samen als woordpaar voorkwamen, als woordparen, die in de leerfase wel samen als woordpaar voorkwamen. Van beide soorten is een gedeelte weer onder G en een gedeelte onder G' geleerd.

De volgende typecodes moeten in de bloktypes opgenomen worden:

Blok1: 11, 55 en 99
Blok2: 34, 67 en 99
Blok3: 11, 55, 34, 68

NB: Op het eerste gezicht lijkt het alsof de aanbiedingen met typecode 11 en 55 staan voor woordparen, die zowel in de leerfase als in de testfase als geheel worden aangeboden. Als je het mechanisme van het programma goed bekijkt blijkt dit juist niet het geval te zijn. Als in de leerfase aan het eerste woord van een woordpaar met typecode 11 een woord uit STIMUL.SCR met volgnr. "i" wordt toegekend, dan wordt aan zijn paargenoot een woord met volgnr. "i+1" toegekend. In de testfase zal hetzelfde woord, met volgnr "i" gekoppeld worden met een woord met volgnr i+JUMP(ITYPE) (JUMP(ITYPE) > 7).

Komen alleen in de aanbiedingstype-code 34 woordtypes 3 en 4 voor, of komen 3, en 4 even vaak in een andere typecode voor, dan komt een paar ab met typecode 34 (a en b zijn volgnummers van woorden in STIMUL.ASC) steeds zowel in de leerfase als in de testfase als paar voor.

§B4: Instructies voor de proefpersonen

In de Pilotstudy en in het herkenningsexperiment krijgen de proefpersonen voorafgaand aan de leer- en test-fase een instructie (zie §5.2 resp. §6.2).

In de Pilotstudy kregen alle proefpersonen voorafgaand aan de leerfase dezelfde instructie. Voorafgaand aan de testfase kregen 4 proefpersonen een neutrale en 4 proefpersonen een Bias-ja instructie. Deze 3 instructies staan hieronder letterlijk weergegeven:
Istructie leerfase:

Je krijgt zo dadelijk 480 woorden na elkaar te zien op het beeldscherm (de helft van de woorden komt tweemaal voor.).
Je taak is deze woorden stuk voor stuk zo goed mogelijk te bestuderen:
- Koncentreer je steeds op het woord, dat aangeboden wordt. (In jezelf herhalen van eerdere woorden heeft geen enkele zin.)
- Een goede strategie is je van elk woord de betekenis te realiseren, en je er iets bij voor te stellen.
- Probeer rustig maar geconcentreerd te werken.
- Richt je op het woord, dat op het scherm staat.

(Neutrale) instructie testfase:

Je krijgt zodadelijk weer 480 woorden na elkaar te zien. Een deel hiervan heb je zojuist bestudeerd (we noemen deze woorden "oud"), de rest heb je niet in de leerfase gezien (Deze woorden noemen we "nieuw").

Je taak is bij elk woord te beslissen, of het "oud" of "nieuw" is. Zodra je je beslissing hebt genomen, druk je zo snel mogelijk het ja-, resp. het nee-knopje in. Probeer bij elk woord zo snel te beslissen als mogelijk is zonder dat dit ten koste gaat van de nauwkeurigheid van deze beslissing.

(Bias-ja) instructie testfase:

Je krijgt zodadelijk weer 480 woorden na elkaar te zien. Een deel hiervan heb je zojuist bestudeerd (we noemen deze woorden "oud"), de rest heb je niet in de leerfase gezien (Deze woorden noemen we "nieuw").

Je taak is bij elk woord te beslissen, of het "oud" of "nieuw" is. Zodra je het gevoel hebt dat je het woord kent uit de leer-fase (ook al is dat gevoel maar vaag) druk je onmiddelijk de ja-knop in (rechts). Alleen als je helmaal niet zo'n gevoel krijgt druk je de nee-knop in.

NB: In de kop boven de beide instructies testfase stond niet aangegeven, dat het een "neutrale" resp. "Bias-ja" instructei was.

In het herkenningsexperiment kregen alle proefpersonen voorafgaand aan de leer-fase dezelfde leerinstructie . Deze verschilde slechts op een punt met de corresponderende in de Pilotstudy; i.p.v. "480 woorden" stond in regel 1 "een groot aantal woorden".
voorafgaand aan de test-fase kregen alle proefpersonen dezelfde Bias-ja instructie. Deze verschilde ook slechts op één punt met de Bias-ja instructie in de Pilotstudy; i.p.v. "480 woorden" stond in regel 1 "een groot aantal woorden".

§B5:bepalen startwaardes modelparameters direct uit de data:

Bepalen startwaardes voor r1, l1, r2, en l2 uit Hx(t,nee):

Er geldt (zie 3.3.14):

$$hx(t,nee) = g(t) \,.\, (1 - F(t)).$$

Hierbij hebben we voor g(t) en F(t) veronderstelt (zie 3.4.2 resp. 3.4.2):

$$g(t) = gamma(r2,l2,g2); \quad F(t) = Gamma(r1,l1,0)$$

In §6.4 is al besproken hoe g2 direct uit de data wordt geschat, zodat we deze parameter verder als een bekende constante beschouwen. Om l1, r2, l2 eenvoudig te kunnen schatten maken we de volgende benadering:

B5.1: $r1 = 1$

Hiermee wordt F(t) exponentieel, zodat we voor hx(t,nee) kunnen schrijven:

$$hx(t,nee) = \exp(-l1.t) \,.\, gamma(r2,l2,g2) =$$

$$\exp(-l1.g1) \times (1 + l1/l2)^{-r1} \,.\, gamma(r2,1/(l1+l2),g2)$$

Hieruit volgt:

B5.3: $hx(t|\ nee) = gamma(r2,1/(l1+l2),g2)$

We vinden nu 3 vergelijkingen:

$$B5.4: Px(nee) = \exp(-l1xg1) \times (1 + l1/l2)^{-r1}$$

$$B5.5: Ex(t|\ nee) = r1/(l1+l2) + g1$$

$$B5.6: Varx(t|\ nee) = r1/((l1+l2)xx2)$$

Px(nee), Ex(t| nee), en Varx(t| nee) zijn eenvoudig direct uit de data te schatten Als schatters nemen we resp. het quocient van het aantal xnee- en het totale aantal x-cues, het eerste moment, resp. het tweede moment bepaald uit de x-nee tijden. We krijgen dan 3 vergelijkingen (B2.1.4 t/m B2.1.6) met 3 onbekenden: r1, l1, en l2, die we eenvoudig kunnen oplossen.

Bepalen startwaardes voor r3, A, B, S1 en S2 uit de data:

r3, A, B, S1, en S2 moeten uit de verdelingen van y-cues (y1 en/of y2) geschat worden. Omdat deze verdelingen veel gecompliceerder zijn, hebben we besloten tot een andere aanpak als hierboven gebruikt bij de schatting van r1, r2, l1, l2. We maken de volgende drie grove veronderstelling:

- Op het tijdstip tmax, waarop de verdelingen Hx(t,ja) t/m Hy2(t,nee) allemaal juist vrijwel gelijk zijn geworden aan 1 zijn de functies V(u) en pi(Sxt/V(t)) ook juist ongeveer 0.
- r3 = 1
- S2 = 2xS1

Hieruit volgt bij benadering:

$$B5.7: S1xtmax/V(tmax) = -4$$

$$B5.8: V(tmax) = 0$$

We komen nu nog één vergelijking te kort om de 5 onbekenden te bepalen. De eerste simultane schattingen voor proefpersoon 10 hebben we echter uitgevoerd met een lineaire FI(t) (zie §6.5), zodat één onbekende vervalt. Hiermee krijgen we een oplosbaar stelsel.

De schattingen voor de modelparameters, die het resultaat zijn van deze "eerste" simultane schatting, zijn daarna voor de meeste proefpersonen gebruikt als startwaardes voor r3, A, B, S1, en S2. Voor sommigen zijn deze aangepast, zodanig dat steeds ongeveer FI(tmax) = 0.

§B6: Uittesten CHIKW2

Alvorens met CHIKW2 de modelparameters uit de data te schatten, hebben we het programma eerst uitgetest op volgens het model gesimuleerde data. Hierbij hebben we gelet op 3 kriteria:

- de gevoeligheid van de schattingen en de residuensom voor de keuze van startwaardes (zie §6.3)
- de kwaliteit van de schattingen, d.w.z.:
  - het verschil tussen de simulatieparameterwaardes en de schattingen
  - het betrouwbaarheidsinterval van de schattingen
  - de correlatie tussen schattingen

    NB: het betrouwbaarheidsinterval en de correlatie worden door MARQUARD bepaald
- de waarde van de totale residuensom; in hoeverre is deze chikwadraat verdeeld?

We hebben voor deze (i.v.m. rekentijd zeer sumiere) test de volgende methode gevolgd: Direct uit de data van een van de proefpersonen (pp 10) hebben we modelparameters geschat. Deze schattingen hebben we gebruikt als simulatieparameterwaardes, waarmee we in totaal 3000 data gesimuleerd hebben, voor zowel x-, y1, als y2-cues 1000. Vervolgens hebben we met CHIKW2 de modelparameters geschat uit deze gesimuleerde data. Hierbij namen we dezelfde stappen als we voor de echte data gepland hadden (zie §5.3: "startwaardes voor verwerkingsprogramma"), met één verschil: als startwaardes gebruikten we niet directe schattingen uit de data (die in dit geval per definitie gelijk zijn aan de simulatie parameters). In plaats daarvan hebben we door variatie van de startwaardes er naar gekeken welke invloed deze hebben op de schattingen.

In tabel B6.1 zijn de resultaten van de schatting van r1, l1, r2, en l 2 uit de data van de x-cues weergeven voor verschillende startwaardes. Bij het lezen van deze tabel en tabel B6.2, en B6.3 is het volgende van belang:

- De startwaardes, waarachter een "!" staat zijn verschillend van de simulatieparameterwaarde.
- Het onderstreepte gedeelte van de schattingen is gelijk aan de schattingen, die verkregen worden met de eerste startwaardes (per tabel)

	r1:	l1:	r2:	l2:	F:
strtw 1	1.5	0.8	1.85	2.2	22.910
schat 1	1.53624500	0.871960172	1.8472645	2.1363351	20.142110
betrw 1	(6.4 %)	(10.3 %)	(5.4 %)	6.7 %)	
strtw 2	1.5	2.0 !	1.85	2.2	934.66
schat 2	1.5362438	0.871958084	1.84712765	2.13633679	20.142110
betrw 2	(6.4 %)	(10.3 %)	(5.4 %)	6.7 %)	
strtw 3	1.5	0.3 !	1.85	2.2	71.673
schat 3	1.5362497	0.871978032	1.84711607	2.13630710	20.142111
betrw 3	(6.4 %)	(10.3 %)	(5.4 %)	6.7 %)	
strtw 4	3.0 !	1.6 !	3.7 !	4.4 !	864.87
schat 4	1.5362438	0.871959649	1.84710731	2.13631357	20.142110
betrw 4	(6.4 %)	(10.3 %)	(5.4 %)	6.7 %)	
strtw 5	1.05 !	0.4 !	1.1 !	1.1 !	1298.1
schat 5	1.53624495	0.871959951	1.84712363	2.13633198	20.142110
betrw 5	(6.4 %)	(10.3 %)	(5.4 %)	6.7 %)	

Tabel B6.1: de resultaten van de schatting van r1, l1, r2, en l 2 uit de data van de x-cues weergegeven voor verschillende startwaardes (strtw 1 t/m 5): de schattingen (schat 1 t/m 5) , de door MARQUARD bepaalde relatieve percentuele standaarddeviaties (betrw 1 t/m 5), en de bijbehorende residuensommen F

Uit tabel B6.1 kan men aflezen, dat zolang de startwaardes niet meer dan een factor 2 afwijken van de simulatieparameterwaardes, de relatieve variatie in de schattingen maximaal een ordegrootte heeft van 10-4. De schattingen voor r1, l1, r2, en l2 wijken 2.4 %, 9 %, 6 %, resp. 3 % af van de simulatiewaardes.

Het aantal klassen, waarin de data zijn onderverdeeld, is 25. Verder is het aantal aangepaste parameters 4, zodat we voor het aantal vrijheidsgraden 25 - 1 - 4 = 20 nemen. Hieruit volgt, dat de residuensom minder als 1 % afwijkt van de verwachtingswaarde, als we uitgaan van een chikwadraatverdeling.

	r3:	B:	S1:	F:
strtw 1	2.0 !	1.0 !	6.0 !	424.2
schat 1	131755676	2.23338050	3.63905378	20.569699
betrw 1	(6.1 %)	(7.3 %)	(9.8 %)	
strtw 2	2.0 !	3.0 !	5.0 !	114.23
schat 2	1.317554	2.23322530	3.63972140	20.569698
betrw 2	(6.1 %)	(7.3 %)	(9.8 %)	

Tabel B6.2: de resultaten van de schatting van r3, B, en S1 uit de data van de y1-cues weergegeven voor verschillende startwaardes (strtw 1 t/m 2): de schattingen (schat 1 t/m 2) , de door MARQUARD bepaalde relatieve percentuele standaarddeviaties (betrw 1 t/m 2), en de bijbehorende residuensommen F

Uit tabel B6.2 kan men aflezen, dat variatie in de startwaardes B en S1 met een factor 3 resp. 5/6 een relatieve afwijking met macimale ordegrootte van E-3 geeft.

De schattingen voor r3, B, en S1 wijken 20%, 1.5 % en 21 % af van de simulatiewaardes. Dit hangt o.i. samen met een hoge correlatie tussen r3 en S1. De door MARQUARD berekende correlatiecoefficient = 0.79.

Het aantal klassen, waarin de data zijn onderverdeeld, is 26. Verder is het aantal aangepaste parameters 3, zodat we voor het aantal vrijheidsgraden 26 - 1 - 3 = 22 nemen. Hieruit volgt, dat de residuensom minder als 7 % lager ligt dan de verwachtingswaarde, als we uitgaan van een chikwadraatverdeling.

§B7: Source-listing van INTAKT en SEQWOR met bijbehorende procedures

Op de volgende pagina's volgt eerst de listing van INTAKT, en bijbehorende procedures, daarna de listing van SEQWOR met bijbehorende procedures (zie §B1).

```
      BLOCK DATA
C
      LOGICAL*1 FILNMS(14)
C
      COMMON /FINAPP/ FILNMS
C
      DATA FILNMS/'D','X','1',':','0','0','0','0','0','0',
     *           '.','S','U','M'/
C
      END
C
C     begin hoofdprogramma
C
      LOGICAL   FIRSQ,INSTRL(4),INSTRT(4),FIRSPP,FIRSES,CHANGE,KLAAR
      LOGICAL*1 FILNMS(14),TYPW01,TYPW10,NSSASC(2)
      INTEGER   NBLOKL,NBLOKT,NDIFL,NDIFT,IDIFL,IDIFT,
     *          NIDMBL,NIDMBT,IIDMBL,IIDMBT,
     *          NTYABL(4),NTYABT(4),ITYABL,ITYABT,NABL(4),NABT(4),
     *          TYPEAL(4,10),TYPEAT(4,10),NPTYAL(4,10),NPTYAT(4,10),
     *          TAL(4),TTL(4),TAT(4),TTT(4),
     *          NSESSY,MINIM,MAXIM,TELWOR,TELWRL,TELWRT,NPTY01,NPTY10,
     *          PL,PC,ENDFIL
C
      COMMON /FINAPP/ FILNMS
      COMMON /EXPAR/ NBLOKL,NBLOKT,NDIFL,NDIFT,
     *               NTYABL,NTYABT,NABL,NABT,
     *               TYPEAL,TYPEAT,NPTYAL,NPTYAT,
     *               TAL,TTL,TAT,TTT,
     *               INSTRL,INSTRT
C
C
      CALL SETPAR(.TRUE.)
      CALL ERPART(1,0,23)
C
C      openen v/d te wijzigen files
C
      OPEN(UNIT=9,ACCESS='DIRECT',MAXREC=1,
     *RECORDSIZE=120,NAME='DX1:LASTEX.SUM',TYPE='OLD',ERR=2)
      FIRSPP=.FALSE.
      GO TO 5
    2 OPEN(UNIT=9,ACCESS='DIRECT',MAXREC=1,INITIALSIZE=1,
     *RECORDSIZE=120,NAME='DX1:LASTEX.SUM',TYPE='NEW')
      FIRSPP=.TRUE.
C
    5 OPEN(UNIT=11,NAME='DX1:PARTAB.ASC',TYPE='NEW')
C
C
    9 CALL ERPART(1,0,23)
      CALL POCURS(1,0)
      CALL PPCODE
      OPEN(UNIT=10,ACCESS='DIRECT',MAXREC=16,
     *RECORDSIZE=120,NAME=FILNMS,TYPE='OLD',ERR=10)
      FIRSES=.FALSE.
      GO TO 30
   10 OPEN(UNIT=10,ACCESS='DIRECT',MAXREC=16,INITIALSIZE=16,
     *RECORDSIZE=120,NAME=FILNMS,TYPE='NEW',ERR=20)
      FIRSES=.TRUE.
      ENDFIL=3
      WRITE(10'16) ENDFIL
      GO TO 30
   20 WRITE(5,*) ' CODE PP IS INCORRECT'
      GO TO 5
   30 CONTINUE
```

```
C
C       het inlezen v/d experimentele parameters v/h laatste experiment
C       v/d pp, resp. het allerlaatste experiment in het array 'parold'
C
      IF (FIRSES) GO TO 70
      READ(10'1) NSESSY
      NSESSY=NSESSY+1
      READ(10'NSESSY) NBLOKL,NBLOKT,NDIFL,NDIFT,
     *                NTYABL,NTYABT,NABL,NABT,
     *                TYPEAL,TYPEAT,NPTYAL,NPTYAT,
     *                TAL,TTL,TAT,TTT,
     *                INSTRL,INSTRT,
     *                MINIM,MAXIM
      GO TO 150
C
 70   CONTINUE
      NSESSY=1
      IF (FIRSPP) GO TO 80
      READ(9'NSESSY)  NBLOKL,NBLOKT,NDIFL,NDIFT,
     *                NTYABL,NTYABT,NABL,NABT,
     *                TYPEAL,TYPEAT,NPTYAL,NPTYAT,
     *                TAL,TTL,TAT,TTT,
     *                INSTRL,INSTRT,
     *                MINIM,MAXIM
      MINIM=0
      MAXIM=0
      GO TO 150
C
 80   CONTINUE
      WRITE(5,*) ' DE WAARDES VAN ALLE EXPERIMENTELE PARAMETERS ZIJN'
      WRITE(5,*) ' "0" GESTELD; U DIENT ZE DUS ZELF EEN WAARDE TOE'
      WRITE(5,*) ' TE KENNEN. KLAAR ? (Y/N)'
      CALL ANSWER(KLAAR)
      GO TO 160
C
C       de eventuele wijzising v/d in 'parold' inselezen experimentele
C       parameters voor het komende experiment door de experimentator
C
 150  CONTINUE
      CALL POCURS(23,0)
      WRITE(5,155)
 155  FORMAT(' WILT U (NOG) VERANDERINGEN AANBRENGEN IN DE',
     *       ' EXP. PARAMETERS (Y/N)?')
      CALL POCURS(23,72)
      CALL ANSWER(CHANGE)
      IF (.NOT.CHANGE) GO TO 200
 160  CONTINUE
C
      CALL ERPART(1,0,23)
      CALL NAMGEG(.TRUE.)
      CALL LEES(.TRUE.)
      CALL NAMGEG(.FALSE.)
      CALL LEES(.FALSE.)
C
      GO TO 150
C
 200  CONTINUE
C
C     berelen aantal te reserveren woorden voor volgende sessie.
C
      TELWOR=0
      TELWRL=0
      TELWRT=0
      NIDMBL=NBLOKL/NDIFL
      NIDMBT=NBLOKT/NDIFT
C
      DO 250 IDIFL=1,NDIFL
        DO 260 ITYABL=1,NTYABL(IDIFL)
          TYPW01=MOD(TYPEAL(IDIFL,ITYABL),10)
          TYPW10=(TYPEAL(IDIFL,ITYABL)-TYPW01)/10
          NPTY01=NPTYAL(IDIFL,ITYABL)
          IF (TYPW01.EQ.2.OR.TYPW01.EQ.7) NPTY01=NPTY01/2
          IF (TYPW10.EQ.0) GO TO 220
          NPTY10=NPTYAL(IDIFL,ITYABL)
          IF (TYPW10.EQ.2.OR.TYPW10.EQ.7) NPTY10=NPTY10/2
```

```
          GO TO 230
 220      NPTY10=0
 230      CONTINUE
          TELWRL=TELWRL+NPTY01+NPTY10
 260    CONTINUE
 250  CONTINUE
C
      DO 290 IDIFT=1,NDIFT
        DO 295 ITYABT=1,NTYABT(IDIFT)
          TYPW01=MOD(TYPEAT(IDIFT,ITYABT),10)
          TYPW10=(TYPEAT(IDIFT,ITYABT)-TYPW01)/10
          IF (TYPW01.EQ.9) TELWRT=TELWRT+NPTYAT(IDIFT,ITYABT)
          IF (TYPW10.EQ.9) TELWRT=TELWRT+NPTYAT(IDIFT,ITYABT)
 295    CONTINUE
 290  CONTINUE
C
      TELWOR=TELWRL*NIDMBL+TELWRT*NIDMBT
      MINIM=MAXIM+1
      MAXIM=MINIM+TELWOR-1
C
C  wegschrijven nieuwe gegevens naar <ppcode>.sum; lastex.sum.
C
      ENCODE(2,299,NSSASC) NSESSY
 299  FORMAT(I2)
      FILNMS(9)=NSSASC(1)
      IF(FILNMS(9).EQ.' ') FILNMS(9)='0'
      FILNMS(10)=NSSASC(2)
C
      WRITE(10'1) NSESSY
C
      WRITE(10'NSESSY+1) NBLOKL,NBLOKT,NDIFL,NDIFT,
     *                   NTYABL,NTYABT,NABL,NABT,
     *                   TYPEAL,TYPEAT,NPTYAL,NPTYAT,
     *                   TAL,TTL,TAT,TTT,
     *                   INSTRL,INSTRT,
     *                   MINIM,MAXIM
C
      WRITE(9'1) NBLOKL,NBLOKT,NDIFL,NDIFT,
     *           NTYABL,NTYABT,NABL,NABT,
     *           TYPEAL,TYPEAT,NPTYAL,NPTYAT,
     *           TAL,TTL,TAT,TTT,
     *           INSTRL,INSTRT,
     *           MINIM,MAXIM,(FILNMS(I),I=5,10)
C
C        wegschrijven gegevens naar partab.asc
C
      NIDMBL=NBLOKL/NDIFL
      NIDMBT=NBLOKT/NDIFT
C
      WRITE(11,320) ((IDIFL,IDIFL=1,NDIFL),IIDMBL=1,NIDMBL),
     *              ((IDIFT+NDIFL,IDIFT=1,NDIFT),IIDMBT=1,NIDMBT)
 320  FORMAT(' SEQ: ',50('L',I1,','))
C
      DO 500 IDIFL=1,NDIFL
        WRITE(11,400) IDIFL,TAL(IDIFL),TTL(IDIFL)
 400    FORMAT(' L',I1,': AAN:',I3,' S/10',/,
     *         '      UIT:',I3,' S/10',/,
     *         '      RES: 0')
 500  CONTINUE
C
      DO 700 IDIFT=1,NDIFT
        WRITE(11,600) IDIFT+NDIFL,TAT(IDIFT),TTT(IDIFT)
 600    FORMAT(' L',I1,': AAN:',I3,' S/10',/,
     *         '      UIT:',I3,' S/10',/,
     *         '      RES: 1')
 700  CONTINUE
C
      CLOSE(UNIT=9)
      CLOSE(UNIT=10)
      CLOSE(UNIT=11)
C
      CALL SETPAR(.FALSE.)
C
      STOP
      END
```

```
TY CODEPP.FOR,CHSCRE.FOR,NAMGEG.FOR,LEES.FOR,LELOFT.FOR
      SUBROUTINE PPCODE
C
      LOGICAL CHANGE
      LOGICAL*1 TEXT(80),ALFA,FILNMS(14)
      COMMON /FINAPP/ FILNMS
C
  10  CONTINUE
C
      DO 20 I=1,80
      TEXT(I)=' '
  20  CONTINUE
      WRITE(5,*) 'CODE PROEFPERSOON?(4 KARAKTERS):',
      WRITE(5,*) 'EERSTE KARAKTER=LETTER.)'
      READ(5,40) INT,(TEXT(I),I=1,INT)
  40  FORMAT(Q,80A1)
C
      IF(INT.EQ.4) GO TO 45
      WRITE(5,*) ' CODE PP IS NIET 4 KARAKTERS LANG.'
      GO TO 10
 45   IF (TEXT(1).GE.65.AND.TEXT(1).LE.90) GO TO 47
      WRITE(5,*) ' 1E KARAKTER=GEEN LETTER'
      GO TO 10
C
  47  CONTINUE
      WRITE(5,48) (TEXT(I),I=1,4)
  48  FORMAT(' DE DOOR U OPGEGEVEN CODE V/D PP. = ',4A1)
      WRITE(5,49)
  49  FORMAT(/,/,' WILT U DEZE NOG VERANDEREN ? (Y/N) ')
      CALL ANSWER(CHANGE)
      IF (CHANGE) GO TO 10
      DO 50 I=1,4
      FILNMS(I+4)=TEXT(I)
  50  CONTINUE
      RETURN
      END

C
C GEBRUIK JE EEN VAN DEZE PROCEDURES IN JE PROGRAMMA DAN MOET JE:
C (1):  AAN HET BEGIN "CALL SETPAR(.TRUE.)", AAN HET EIND "CALL
C       SETPAR(.FALSE.)" OPNEMEN
C (2): HET PROGRAMMA LINKEN MET "CHDATY.OBJ"
C
C
      SUBROUTINE SETPAR(PARSET)
      LOGICAL*1 ESC
      LOGICAL PARSET
      COMMON /SCREEN/ ESC
      ESC='033
      IF (.NOT.PARSET) GO TO 200
      WRITE(5,100) ESC,ESC,ESC,ESC
 100  FORMAT(' ',A1,'PrSM3',A1,'\',A1,'[2J',A1,'[H')
      GO TO 300
 200  CONTINUE
      WRITE(5,150) ESC,ESC,ESC,ESC
 150  FORMAT(' ',A1,'PrSM2',A1,'\',A1,'8',A1,'[20B')
 300  CONTINUE
      RETURN
      END
C             END SETPAR
C
      SUBROUTINE POCURS(PL,PC)
      INTEGER PL,PC
      LOGICAL*1 ESC
      COMMON /SCREEN/ ESC
      IF (PL.LT.10.AND.PC.LT.10) GO TO 10
      IF (PL.LT.10.AND.PC.GE.10) GO TO 20
```

```
      IF (PL.GE.10.AND.PC.LT.10) GO TO 30
      IF (PL.GE.10.AND.PC.GE.10) GO TO 40
   10 WRITE(5,100) ESC,PL,PC
  100 FORMAT($' ',A1,'[',I1,';',I1,'H')
      GO TO 200
   20 WRITE(5,120) ESC,PL,PC
  120 FORMAT($' ',A1,'[',I1,';',I2,'H')
      GO TO 200
   30 WRITE(5,130) ESC,PL,PC
  130 FORMAT($' ',A1,'[',I2,';',I1,'H')
      GO TO 200
   40 WRITE(5,140) ESC,PL,PC
  140 FORMAT($' ',A1,'[',I2,';',I2,'H')
  200 CONTINUE
      RETURN
      END
C             END POCURS
C
C
      SUBROUTINE ERPART(PL,PC,NLINE)
      INTEGER PL,PC,NLINE
      LOGICAL*1 ESC
      COMMON /SCREEN/ ESC
      DO 200 I=1,NLINE
      CALL POCURS(PL+I-1,PC)
      WRITE(5,100) ESC
  100 FORMAT(' ',A1,'[K')
  200 CONTINUE
      RETURN
      END
C             END ERPART
C
      SUBROUTINE UITLEG(PL,PC)
      INTEGER PL,PC
      LOGICAL*1 ESC
      COMMON /SCREEN/ ESC
      CALL SKLEUR(33)
      CALL POCURS(19,0)
      WRITE(5,100)
  100 FORMAT(
     *' REGEL NIET VERANDEREN: - DRUK METEEN <RETURN> IN!!!',/,
     *' REGEL WEL VERANDEREN: - TYP EEN NIEUWE REGEL IN; ',/,
     *' (CORRECTIETIP: <DELETE> WIST DE LAATSTE KARAKTER V/E REGEL UIT)'
     *,/,' - DRUK VERVOLGENS <RETURN> IN.')
      CALL SKLEUR(37)
      CALL POCURS(PL,PC)
      RETURN
      END
C
      SUBROUTINE SKLEUR(KLEUR)
      INTEGER KLEUR
      LOGICAL*1 ESC
      COMMON /SCREEN/ ESC
      WRITE(5,50) ESC,KLEUR
   50 FORMAT(' ',A1,'[',I2,'m')
      RETURN
      END
C
C
      SUBROUTINE ANSWER(LOANSW)
      LOGICAL LOANSW
      LOGICAL*1 ANSW
   10 READ(5,11) ANSW
   11 FORMAT(A1)
      IF (ANSW.EQ.'Y') GO TO 15
      IF (ANSW.EQ.'N') GO TO 13
      GO TO 17
   13 LOANSW=.FALSE.
      GO TO 19
   15 LOANSW=.TRUE.
      GO TO 19
   17 WRITE(5,*) ' ANTWOORD: "Y <RETURN>", OF "N <RETURN>" !!!'
```

```
      GO TO 10
   19 CONTINUE
      RETURN
      END

      SUBROUTINE NAMGEG(FIRSQ)
C
C
      LOGICAL   FIRSQ,INSTRL(4),INSTRT(4)
      INTEGER   NBLOKL,NBLOKT,NDIFL,NDIFT,IDIFL,IDIFT,
     *          NTYABL(4),NTYABT(4),ITYABL,ITYABT,NABL(4),NABT(4),
     *          TYPEAL(4,10),TYPEAT(4,10),NPTYAL(4,10),NPTYAT(4,10),
     *          TAL(4),TTL(4),TAT(4),TTT(4),
     *          PL,PC
      LOGICAL*1 LINST
C
C
      COMMON /EXPAR/ NBLOKL,NBLOKT,NDIFL,NDIFT,
     *               NTYABL,NTYABT,NABL,NABT,
     *               TYPEAL,TYPEAT,NPTYAL,NPTYAT,
     *               TAL,TTL,TAT,TTT,
     *               INSTRL,INSTRT
C
C
      IF (.NOT.FIRSQ) GO TO 100
C
      CALL ERPART(0,0,23)
C
C         regel 1 t/m 3:
C
      CALL SKLEUR(33)
      CALL POCURS(0,0)
      WRITE(5,10)
   10 FORMAT(' TOTAAL AANTAL BLOKKEN,AANTAL VERSCHILLENDE BLOKKEN,',
     *       ' AANTAL TYPES PER BLOK,',/,
     *       ' LEERFASE:',/,
     *       ' TESTFASE:')
      CALL SKLEUR(32)
      CALL POCURS(2,10)
      WRITE(5,20) NBLOKL,NDIFL,(NTYABL(IDIFL),IDIFL=1,NDIFL)
      CALL POCURS(3,10)
      WRITE(5,20) NBLOKT,NDIFT,(NTYABT(IDIFT),IDIFT=1,NDIFT)
   20 FORMAT(' ',I2,',',I1,',',4(I2,',',I2,','))
      CALL SKLEUR(37)
      GO TO 500
C
C         regel 4 t/m 15:
C
C         leerfase:
C
  100 CONTINUE
      CALL SKLEUR(33)
      CALL POCURS(4,0)
      WRITE(5,105)
  105 FORMAT(' LEERFASE: PER BLOK: INSTRUCTIE? (Y/N), AANBIEDINGS-',
     *       ' TUSSEN-TIJD,',/,
     *       ' (PER AANBIEDINGSTYPE: CODE,AANTAL)')
C
      PL=6
      PC=0
      DO 199 IDIFL=1,NDIFL
        IF (INSTRL(IDIFL)) LINST='Y'
        IF (.NOT.INSTRL(IDIFL)) LINST='N'
        CALL SKLEUR(33)
        CALL POCURS(PL,PC)
        WRITE(5,107) IDIFL
        CALL SKLEUR(32)
        CALL POCURS(PL,8)
        WRITE(5,110) LINST,TAL(IDIFL),TTL(IDIFL),
     *               ((TYPEAL(IDIFL,ITYABL),NPTYAL(IDIFL,ITYABL)),
     *                ITYABL=1,NTYABL(IDIFL))
        PL=PL+2
```

```
 199  CONTINUE
C
C     testfase
C
      CALL SKLEUR(33)
      CALL POCURS(14,0)
      WRITE(5,205)
 205  FORMAT(' TESTFASE: PER BLOK: (ZIE LEERFASE!!!)')
C
      PL=15
      PC=0
      DO 299 IDIFT=1,NDIFT
        IF (INSTRT(IDIFT)) LINST='Y'
        IF (.NOT.INSTRT(IDIFT)) LINST='N'
        CALL SKLEUR(33)
        CALL POCURS(PL,PC)
        WRITE(5,107) IDIFT
        CALL SKLEUR(32)
        CALL POCURS(PL,8)
        WRITE(5,110) LINST,TAT(IDIFT),TTT(IDIFT),
     *               ((TYPEAT(IDIFT,ITYABT),NPTYAT(IDIFT,ITYABT)),
     *                ITYABT=1,NTYABT(IDIFT))
 107    FORMAT($' BLOK',I1,': ')
 110    FORMAT($' ',A1,',',2(I2,','),10(I2,',',I3,','))
        PL=PL+2
 299  CONTINUE
C
C
 500  CONTINUE
      CALL SKLEUR(37)
C
      RETURN
      END

      SUBROUTINE LEES(FIRSQ)
C
      INTEGER   NBLOKL,NBLOKT,NDIFL,NDIFT,IDIFL,IDIFT,
     *          NTYABL(4),NTYABT(4),ITYABL,ITYABT,NABL(4),NABT(4),
     *          TYPEAL(4,10),TYPEAT(4,10),NPTYAL(4,10),NPTYAT(4,10),
     *          TAL(4),TTL(4),TAT(4),TTT(4),
     *          PL,PC,IREGEL,IPARNW,PARNEW(15),INT
      LOGICAL   INSTRL(4),INSTRT(4),FIRSQ
C
C
      COMMON /EXPAR/ NBLOKL,NBLOKT,NDIFL,NDIFT,
     *               NTYABL,NTYABT,NABL,NABT,
     *               TYPEAL,TYPEAT,NPTYAL,NPTYAT,
     *               TAL,TTL,TAT,TTT,
     *               INSTRL,INSTRT
C
      IF (.NOT.FIRSQ) GO TO 200
C
C        eerste 2 regels worden ingelezen en getest.
C
      DO 100 IREGEL=1,2
C
        CALL ERPART(23,0,1)
C
   5    CONTINUE
        DO 7 IPARNW=1,15
          PARNEW(IPARNW)=0
   7    CONTINUE
        PL=IREGEL+1
        CALL POCURS(PL,30)
C
        READ(5,10,ERR=1100) INT,(PARNEW(IPARNW),IPARNW=1,15)
  10    FORMAT(Q,15I7)
C
        IF (INT.EQ.0) GO TO 100
C
C
C    regel 'I' wordt getest.
```

```
C
      DO 30 IPARNW=1,15
        IF (PARNEW(IPARNW).EQ.0) GO TO 40
 30   CONTINUE
 40   IPARNW=IPARNW-1
C
      IF (PARNEW(1).GT.50) GO TO 1110
      IF (PARNEW(2).GT.4) GO TO 1120
      IF (MOD(PARNEW(1),PARNEW(2)).NE.0) GO TO 1130
      IF ((IPARNW-2).NE.PARNEW(2)) GO TO 1140
C
      DO 50 I=1,PARNEW(2)
        IF (PARNEW(I+2).GT.9) GO TO 1150
 50   CONTINUE
C
      GO TO 90
C
C     foutmeldingen behorende bij regel 'I'
C
1100  CONTINUE
      CALL SKLEUR(31)
      CALL POCURS(23,0)
      WRITE(5,*) ' FOUT: KARAKTER IS GEEN LETTER; '
      GO TO 1200
1110  CONTINUE
      CALL SKLEUR(31)
      CALL POCURS(23,0)
      WRITE(5,*) ' FOUT: TOTAAL AANTAL BLOKKEN(NBLOK) > 50'
      GO TO 1200
1120  CONTINUE
      CALL SKLEUR(31)
      CALL POCURS(23,0)
      WRITE(5,*) ' FOUT: AANTAL VERSCHILLENDE BLOKKEN(NDIF) > 4'
      GO TO 1200
1130  CONTINUE
      CALL SKLEUR(31)
      CALL POCURS(23,0)
      WRITE(5,*) ' FOUT: MOD(NBLOK,NDIF) IS NIET "0" '
      GO TO 1200
1140  CONTINUE
      CALL SKLEUR(31)
      CALL POCURS(23,0)
      WRITE(5,*) ' FOUT: TE VEEL OF WEINIG AANTALLEN TYPES PER BLOK'
      GO TO 1200
1150  CONTINUE
      CALL SKLEUR(31)
      CALL POCURS(23,0)
      WRITE(5,*) ' FOUT: AANTAL TYPES PER BLOK > 9;'
C
1200  CONTINUE
      CALL SKLEUR(37)
      CALL ERPART(PL,30,1)
      GO TO 5
C
C     de zojuist in PARNEW ingelezen en geteste experimentele parameters
C     corresponderend met regel 'I' worden opgeslagen.
C
 90     CONTINUE
        IF (IREGEL.EQ.2) GO TO 95
C
        NBLOKL=PARNEW(1)
        NDIFL=PARNEW(2)
        DO 92 IDIFL=1,NDIFL
          NTYABL(IDIFL)=PARNEW(2+IDIFL)
 92     CONTINUE
        GO TO 99
C
 95     CONTINUE
        NBLOKT=PARNEW(1)
        NDIFT=PARNEW(2)
        DO 97 IDIFT=1,NDIFT
          NTYABT(IDIFT)=PARNEW(2+IDIFT)
```

```
  97     CONTINUE
  99     CONTINUE
C
 100   CONTINUE
       GO TO 300
C
C   regel 6-9 en 12-15 worden ingelezen en getest.
C
 200   CONTINUE
       CALL LELOFT(NDIFL,NTYABL,NABL,5,INSTRL,TAL,TTL,TYPEAL,NPTYAL)
       CALL LELOFT(NDIFT,NTYABT,NABT,14,INSTRT,TAT,TTT,TYPEAT,NPTYAT)
C
 300   CONTINUE
C
       RETURN
       END

       SUBROUTINE LELOFT(NDIF,NTYAB,NAB,PLO,INSTR,TA,TT,TYPEA,NPTYA)
C
       INTEGER   NDIF,IDIF,
      *          NTYAB(4),ITYAB,NAB(4),
      *          TYPEA(4,10),NPTYA(4,10),
      *          TA(4),TT(4),
      *          PL,PC,PLO,IPARNW,PARNEW(22),INT,DUM1,DUM2,SUMNPT
       LOGICAL   INSTR(4)
       LOGICAL*1 LINST,DKOM
C
C
C
 2000  CONTINUE
C
       PL=PLO
C
       DO 2500 IDIF=1,NDIF
C
 2005    CONTINUE
C
         CALL ERPART(23,0,1)
         PL=PL+2
         PC=0
 2010    CONTINUE
C
 2015    CALL POCURS(PL,PC)
         DO 2020 IPARNW=1,22
           PARNEW(IPARNW)=0
 2020    CONTINUE
C
         READ(5,2022,ERR=3100) INT,LINST,DKOM,
     *                         (PARNEW(IPARNW),IPARNW=1,22)
 2022    FORMAT(Q,2A1,22I7)
C
         IF (INT.EQ.0) GO TO 2500
C
C
         DO 2025 IPARNW=1,22
           IF (PARNEW(IPARNW).EQ.0) GO TO 2027
 2025    CONTINUE
 2027    CONTINUE
C
C    nieuwe gegevens worden getest.
C
         IPARNW=IPARNW-1
         DUM1=IPARNW-2
         DUM2=2*NTYAB(IDIF)
         IF (DUM1.NE.DUM2) GO TO 3200
 2028    CONTINUE
C
         IF ((LINST.NE.'Y').AND.(LINST.NE.'N')) GO TO 3250
C
         DO 2110 IPARNW=1,2
           IF ((PARNEW(IPARNW).LT.1).OR.(PARNEW(IPARNW).GT.99))
     *       GO TO 3260
```

```
 2110   CONTINUE
C
       DO 2120 ITYAB=1,NTYAB(IDIF)
C
         TYCO=PARNEW(1+2*ITYAB)
         IF ((TYCO.LT.1).OR.(TYCO.GT.99)) GO TO 3270
C
         NPTYCO=PARNEW(2+2*ITYAB)
         IF ((NPTYCO.LT.1).OR.(NPTYCO.GT.999)) GO TO 3280
         IF ((MOD(NPTYCO,7).EQ.0).OR.(MOD(NPTYCO,11).EQ.0).OR.
     *       (MOD(NPTYCO,13).EQ.0).OR.(MOD(NPTYCO,17).EQ.0))
     *       GO TO 3290
         IF (MOD(NPTYCO,2).NE.0) GO TO 3295
C
 2120   CONTINUE
C
C    parameters krijgen de nieuwe waarden.
C
       IF (LINST.EQ.'Y') INSTR(IDIF)=.TRUE.
       IF (LINST.EQ.'N') INSTR(IDIF)=.FALSE.
       TA(IDIF)=PARNEW(1)
       TT(IDIF)=PARNEW(2)
C
       SUMNPT=0
C
       DO 2030 ITYAB=1,NTYAB(IDIF)
C
         DUM1=ITYAB*2+1
         TYPEA(IDIF,ITYAB)=PARNEW(DUM1)
         DUM2=ITYAB*2+2
         NPTYA(IDIF,ITYAB)=PARNEW(DUM2)
         SUMNPT=SUMNPT+PARNEW(DUM2)
C
 2030   CONTINUE
C
       NAB(IDIF)=SUMNPT
C
C
       GO TO 2500
C
 3100   CONTINUE
       CALL SKLEUR(31)
       CALL POCURS(23,0)
       WRITE(5,*) ' FOUT: KARAKTER IS  LETTER NOCH CIJFER '
       GO TO 3400
 3200   CONTINUE
       CALL SKLEUR(31)
       CALL POCURS(23,0)
       WRITE(5,*) ' FOUT: VERKEERD AANTAL INVOERGEGEVENS'
       GO TO 3400
C
 3250   CONTINUE
       CALL SKLEUR(31)
       CALL POCURS(23,0)
       WRITE(5,*) ' FOUT: ANTWOORD OP "INSTRUCTIE?" NIET "Y" OF "N"'
       GO TO 3400
C
 3260   CONTINUE
       CALL SKLEUR(31)
       CALL POCURS(23,0)
       WRITE(5,*) ' FOUT: AANBIEDINGS- OF TUSSEN-TYD TE GROOT/KLEIN'
       GO TO 3400
C
 3270   CONTINUE
       CALL SKLEUR(31)
       CALL POCURS(23,0)
       WRITE(5,*) ' FOUT: TYPECODE LIGT NIET TUSSEN 1 EN 99'
       GO TO 3400
C
 3280   CONTINUE
       CALL SKLEUR(31)
       CALL POCURS(23,0)
```

```
        WRITE(5,*) ' FOUT: AANTAL PER TYPE LIGT NIET TUSSEN 1 EN 999'
        GO TO 3400
C
 3290   CONTINUE
        CALL SKLEUR(31)
        CALL POCURS(23,0)
        WRITE(5,*) ' FOUT: AANTAL PER TYPE DEELBAAR DOOR 7,11,13, OF 17'
        GO TO 3400
C
 3295   CONTINUE
        CALL SKLEUR(31)
        CALL POCURS(23,0)
        WRITE(5,*) ' FOUT: AANTAL PER TYPE NIET DEELBAAR DOOR 2'
        GO TO 3400
C
 3400   CONTINUE
        CALL SKLEUR(37)
        CALL ERPART(PL,PC,1)
        GO TO 2015
C
 2500 CONTINUE
C
C
      RETURN
      END
```

```
 TY LSEQWO.F\F\X\X\COM
LINK SEQWOR,SBSEQW,SCHRYF,SEQAAN,TOEWYS
RUN SEQWOR


.TY SEQWOR.FOR,SBSEQW.FOR,SCHRYF.FOR,SEQAAN.FOR,TOEWYS.FOR
      LOGICAL   INSTRL(4),INSTRT(4)
      LOGICAL*1 CODEPP(6),VOLGOR(1200)
      INTEGER   NBLOKL,NBLOKT,NDIFL,NDIFT,NIDEML,NIDEMT,
     *          NTYABL(4),NTYABT(4),NABL(4),NABT(4),
     *          TYPEAL(4,10),TYPEAT(4,10),NPTYAL(4,10),NPTYAT(4,10),
     *          TAL(4),TTL(4),TAT(4),TTT(4),MINIM,MAXIM,
     *          NEXTWL(10),NEXTWT(10),JUMP(10),
     *          NTYW,TYPEW(10),NPTYPW(10),MIMATY(10,3)
C
      COMMON /JUMPER/ JUMP
      COMMON /VOLGOR/ VOLGOR
      COMMON /CODEPP/ CODEPP
      COMMON /DIFLOT/ NEXTWL,NEXTWT
      COMMON /IDMLET/ NTYW,TYPEW,NPTYPW,MIMATY
      COMMON /EXPAR/  NBLOKL,NBLOKT,NDIFL,NDIFT,NIDEML,NIDEMT,
     *                NTYABL,NTYABT,NABL,NABT,
     *                TYPEAL,TYPEAT,NPTYAL,NPTYAT,
     *                TAL,TTL,TAT,TTT,
     *                INSTRL,INSTRT,MINIM,MAXIM
C
      DATA JUMP/1,1,1,1,1,1,1,1,1,1/
C
C      in dit stuk wordt:
C            - het file 'lastex.sum' geopend
C            - de experimentele parameters uit dit file gelezern
C            - het file 'lastex.sum' gesloten
C
      OPEN(UNIT=9,ACCESS='DIRECT',MAXREC=1,
     *RECORDSIZE=120,NAME='DX1:LASTEX.SUM',TYPE='OLD')
C
      OPEN(UNIT=10,NAME='DX1:INPSY.ASC',TYPE='NEW')
C
      READ(9'1)       NBLOKL,NBLOKT,NDIFL,NDIFT,
     *                NTYABL,NTYABT,NABL,NABT,
     *                TYPEAL,TYPEAT,NPTYAL,NPTYAT,
     *                TAL,TTL,TAT,TTT,
     *                INSTRL,INSTRT,
     *                MINIM,MAXIM,CODEPP
C
      CLOSE(UNIT=9)
C
      WRITE(10,5) CODEPP
 5    FORMAT(6A1)
C
C   voor beschrijving functie van dit stuk zie procedurebeschrijvingen.
C
      CALL TOEWYS
C
C
      CALL SEQAAN(NBLOKL,NDIFL,NIDEML,NABL,NTYABL,TYPEAL,NPTYAL)
      CALL SCHRYF(NIDEML,NDIFL,NABL,-2,INSTRL,NEXTWL)
C
C
      CALL JUMPTE
      CALL SEQAAN(NBLOKT,NDIFT,NIDEMT,NABT,NTYABT,TYPEAT,NPTYAT)
      CALL SCHRYF(NIDEMT,NDIFT,NABT,-3,INSTRT,NEXTWT)
C
C
      CLOSE(UNIT=10)
C
      STOP
      END
```

```
      SUBROUTINE ZET(TYPEA,NPTYPA)
C
      INTEGER   TYPEA,NPTYPA,IPTYPA,MIINVO,MAINVO,IRA
      LOGICAL*1 VOLGOR(1200)
C
      COMMON /VOLGOR/ VOLGOR
      COMMON /ZETBRE/ MIINVO,MAINVO
C
      DO 1000 IPTYPA=1,NPTYPA
C
        CALL RANDOM(MIINVO-1,MAINVO,IRA,IPTYPA)
C
  150   CONTINUE
          IF (VOLGOR(IRA).EQ.0) GO TO 200
          IRA=IRA+11
          IF (IRA.GT.MAINVO) IRA=IRA-MAINVO+MIINVO-1
        GO TO 150
  200   CONTINUE
        VOLGOR(IRA)=TYPEA
 1000 CONTINUE
C
      RETURN
      END
C           zet: zet nptypa maal de waarde 'typea' in het array
C           volgor op random gekozen plaatsen, die daarvoor nog
C           de waarde '0' hadden.
C
C
C
        SUBROUTINE RANDOM(IMIN,IMAX,IRAND,KEER)
        COMMON /RAND/ IRAN,JRAN
C
        ATIME=SECNDS(0)
        BTIME=AMOD(ATIME,100.)
        ITIME=IFIX(BTIME*300)
C       IF(IABS(IRAN).GE.30000) IRAN=1
C       IF(IABS(JRAN).GE.29000) JRAN=0
        IRAN=ITIME+KEER

        DO 10,I=1,20
                IRANX=IRAN
                RAND=RAN(IRANX,IRAN)
                IRAN=IRAN+1
                JRAN=JRAN-1
   10   CONTINUE
C
        IRAND=IFIX(RAN(IRANX,JRAN)*FLOAT(IMAX-IMIN)+FLOAT(IMIN))+1
C
        RETURN
        END
C
C       THIS ROUTINE GENERATES A RANDOM NUMBER IN THE RANGE
C       IMIN+1 ---> IMAX
C
C
C
      SUBROUTINE BREEK
C
      INTEGER   MIINVO,MAINVO,TEL,TELRY,TYPE,MAXRY(2,3)
      LOGICAL*1 VOLGOR(1200)
C
      COMMON /VOLGOR/ VOLGOR
      COMMON /ZETBRE/ MIINVO,MAINVO
C
    5 DO 6 I=1,2
        DO 6 J=1,3
          MAXRY(I,J)=0
    6 CONTINUE
      TEL=MIINVO
C
   10 CONTINUE
      IF (TEL.GE.MAINVO) GO TO 1000
```

```
        TELRY=0
        TYPE=VOLGOR(TEL)
C
 100    IF (TEL.EQ.MAINVO+1) GO TO 300
 150      IF (VOLGOR(TEL)-TYPE) 300,200,300
 200      TEL=TEL+1
          TELRY=TELRY+1
        GO TO 100
C
 300    IF (TELRY-MAXRY(1,1)) 400,500,600
 400    IF (TELRY-MAXRY(2,1)) 10,10,420
C
 420    IF (TYPE-MAXRY(1,3)) 425,10,425
 425    MAXRY(2,1)=TELRY
        MAXRY(2,2)=TEL-1-TELRY/2
        MAXRY(2,3)=TYPE
        GO TO 10
C
 500    IF (TYPE-MAXRY(1,3)) 425,10,425
 600    IF (TYPE-MAXRY(1,3)) 620,610,620
 610    MAXRY(1,1)=TELRY
        MAXRY(1,2)=TEL-1-TELRY/2
        MAXRY(1,3)=TYPE
        GO TO 10
C
 620    MAXRY(2,1)=MAXRY(1,1)
        MAXRY(2,2)=MAXRY(1,2)
        MAXRY(2,3)=MAXRY(1,3)
        MAXRY(1,1)=TELRY
        MAXRY(1,2)=TEL-1-TELRY/2
        MAXRY(1,3)=TYPE
        GO TO 10
C
 1000 CONTINUE
C
      IF (MAXRY(2,1)-3) 2000,2000,1300
 1300 TYPE=VOLGOR(MAXRY(1,2))
      VOLGOR(MAXRY(1,2))=VOLGOR(MAXRY(2,2))
      VOLGOR(MAXRY(2,2))=TYPE
      GO TO 5
C
 2000 CONTINUE
C
      RETURN
      END
C           breek: deze subroutine onderzoekt of er rijtjes met dezelfde
C           waarde in het array volgor voorkomen, en breekt zulke rijtjes
C           in tweeen door verwisseling van de waardes van de resp.
C           middens van het langste rijtje en het op een na langste rijtje
C           met een verschillende waarde.
C
C
C
      SUBROUTINE JUMPTE
C
      INTEGER NTYW,ITYW,NPTYPW(10),TYPEW(10),MIMATY(10,3),JUMP(10),
     *        PRIBUF(5),PRIBKW(5),TEL
C
      COMMON /JUMPER/   JUMP
      COMMON /IDMLET/ NTYW,TYPEW,NPTYPW,MIMATY
C
      DATA PRIBUF/7,11,13,17,23/,PRIBKW/49,121,169,289,529/
C
      DO 1000 ITYW=1,NTYW
C
        DO 100 TEL=1,5
          IF (PRIBKW(TEL).GT.NPTYPW(ITYW)) GO TO 200
 100    CONTINUE
        TEL=5
C
 200    CONTINUE
        IF (TEL.EQ.1.OR.
```

```
     *     (PRIBKW(TEL)-NPTYPW(ITYW)).LT.(NPTYPW(ITYW)-PRIBKW(TEL-1)))
     *     GO TO 300
           JUMP(ITYW)=PRIBUF(TEL-1)
           GO TO 400
 300       JUMP(ITYW)=PRIBUF(TEL)
 400       CONTINUE
C
 1000 CONTINUE
      RETURN
      END
C           Jump: bepaalt a.h.v nptypw bij elk woordtype de meest geschikte
C                 spronggrootte.

      SUBROUTINE SCHRYF(NIDEMB,NDIF,NAB,STCOFA,INSTRU,NEXTW)
C
      INTEGER   NIDEMB,NDIF,IIDEMB,IDIF,IDIF1,
     *          NAB(4),IAB,IVOLG,NTYW,ITYW,JUMP(10),
     *          NEXTW(10),DNEXTW,TYPEW(10),NPTYPW(10),
     *          STCOFA,STCOBU(5),STOPCO(2),TYPE,WORD(2),I,MIMATY(10,3)
      LOGICAL*1 VOLGOR(1200),CODEPP(6)
      LOGICAL   INSTRU(4)
C
      COMMON /JUMPER/ JUMP
      COMMON /VOLGOR/ VOLGOR
      COMMON /CODEPP/ CODEPP
      COMMON /IDMLET/ NTYW,TYPEW,NPTYPW,MIMATY
C
C
      STOPCO(1)=1
      DO 50 IDIF=1,NDIF
        STCOBU(IDIF)=-1
        IDIF1=MOD(IDIF,NDIF)+1
        IF (INSTRU(IDIF1)) STCOBU(IDIF)=-2
 50   CONTINUE
C
      IVOLG=0
C
      DO 1000 IIDEMB=0,NIDEMB-1
C
        IF (IIDEMB.EQ.NIDEMB-1) STCOBU(NDIF)=STCOFA
C
        DO 500 IDIF=1,NDIF
C
          STOPCO(2)=0
C
          DO 590 IAB=1,NAB(IDIF)
C
            IVOLG=IVOLG+1
            TYPE=VOLGOR(IVOLG)
            WORD(2)=MOD(TYPE,10)
            WORD(1)=(TYPE-WORD(2))/10
            IF (IAB.EQ.NAB(IDIF)) STOPCO(2)=STCOBU(IDIF)
C
            DO 591 I=1,2
C
              IF (WORD(I).EQ.0) GO TO 591
C
              DO 592 ITYW=1,NTYW-1
                IF (TYPEW(ITYW).EQ.WORD(I)) GO TO 593
 592          CONTINUE
 593          CONTINUE
C
              WRITE(10,594) WORD(I),NEXTW(ITYW),STOPCO(I)
 594          FORMAT(' ',I1,X,I4,X,I2)
C
              DNEXTW=NEXTW(ITYW)+JUMP(ITYW)
              IF (DNEXTW.GT.MIMATY(ITYW,2))
     *            DNEXTW=DNEXTW-MIMATY(ITYW,3)
              NEXTW(ITYW)=DNEXTW
C
 591        CONTINUE
C
```

```
 590      CONTINUE
C
 500    CONTINUE
C
 1000 CONTINUE
C
C
      RETURN
      END

      SUBROUTINE SEQAAN(NBLOK,NDIFB,NIDEMB,NAB,NTYAB,TYPEA,NPTYPA)
C
      INTEGER   MIINVO,MAINVO,INDVOL,
     *          NBLOK,NDIFB,NIDEMB,IBLOK,IDIFB,IIDEMB,
     *          NAB(4),NTYAB(4),ITYAB,TYPEA(4,10),NPTYPA(4,10)
      LOGICAL*1 VOLGOR(1200)
C
      COMMON /ZETBRE/ MIINVO,MAINVO
      COMMON /VOLGOR/ VOLGOR
C
C
      MIINVO=0
      MAINVO=0
      DO 10 INDVOL=1,1200
        VOLGOR(INDVOL)=0
 10   CONTINUE
C
      DO 1000 IIDEMB=0,NIDEMB-1
C
        DO 500 IDIFB=1,NDIFB
C
          MIINVO=MAINVO+1
          MAINVO=MIINVO+NAB(IDIFB)-1
C
          IF (NTYAB(IDIFB).EQ.1) GO TO 200
C
          DO 100 ITYAB=1,NTYAB(IDIFB)-1
            CALL ZET(TYPEA(IDIFB,ITYAB),NPTYPA(IDIFB,ITYAB))
 100      CONTINUE
C
 200      CONTINUE
          DO 300 INDVOL=MIINVO,MAINVO
            IF (VOLGOR(INDVOL).EQ.0)
     *          VOLGOR(INDVOL)=TYPEA(IDIFB,NTYAB(IDIFB))
 300      CONTINUE
C
          IF (NTYAB(IDIFB).EQ.0) GO TO 400
          CALL BREEK
C
 400      CONTINUE
C
 500    CONTINUE
C
 1000 CONTINUE
C
      RETURN
      END

      SUBROUTINE TOEWYS
C
      LOGICAL   INSTRL(4),INSTRT(4)
      INTEGER   NBLOKL,NBLOKT,
     *          NDIFL,NDIFT,IDIFL,IDIFT,NIDEML,NIDEMT,
     *          NTYABL(4),NTYABT(4),ITYABL,ITYABT,
     *          NABL(4),NABT(4),
     *          TYPEAL(4,10),TYPEAT(4,10),NPTYAL(4,10),NPTYAT(4,10),
     *          TAL(4),TTL(4),TAT(4),TTT(4),MINIM,MAXIM,
     *          TYPE,WORD(2),NWORD,TYPEW(10),NPTYPW(10),ITYW,NTY9,
     *          MIMATY(10,3),NEXTWL(10),NEXTWT(10)
C
      COMMON /IDMLET/ NTYW,TYPEW,NPTYPW,MIMATY
      COMMON /DIELDT/ NEXTWL,NEXTWT
```

```
      COMMON /EXPAR/  NBLOKL,NBLOKT,NDIFL,NDIFT,NIDEML,NIDEMT,
     *                NTYABL,NTYABT,NABL,NABT,
     *                TYPEAL,TYPEAT,NPTYAL,NPTYAT,
     *                TAL,TTL,TAT,TTT,
     *                INSTRL,INSTRT,MINIM,MAXIM
C
      NIDEML=NBLOKL/NDIFL
      NIDEMT=NBLOKT/NDIFT
C
      NTYW=0
      DO 10 ITYW=1,10
        TYPEW(ITYW)=0
        NPTYPW(ITYW)=0
        MIMATY(ITYW,1)=0
        MIMATY(ITYW,2)=0
        MIMATY(ITYW,3)=0
        NEXTWL(ITYW)=0
        NEXTWT(ITYW)=0
 10   CONTINUE
C
      DO 1000 IDIFL=1,NDIFL
C
        DO 100 ITYABL=1,NTYABL(IDIFL)
C
          TYPE=TYPEAL(IDIFL,ITYABL)
          WORD(2)=MOD(TYPE,10)
          WORD(1)=(TYPE-WORD(2))/10
C
          DO 150 I=1,2
C
            IF (WORD(I).EQ.0) GO TO 150
            NWORD=NPTYAL(IDIFL,ITYABL)
            IF ((WORD(I).EQ.2).OR.(WORD(I).EQ.7)) NWORD=NWORD/2
C
            DO 155 ITYW=1,9
              IF (TYPEW(ITYW).EQ.WORD(I)) GO TO 157
 155        CONTINUE
            NTYW=NTYW+1
            NPTYPW(NTYW)=NWORD
            TYPEW(NTYW)=WORD(I)
            GO TO 150
 157        CONTINUE
            NPTYPW(ITYW)=NPTYPW(ITYW)+NWORD
C
 150      CONTINUE
C
 100    CONTINUE
C
 1000 CONTINUE
C
      DO 1100 ITYW=1,NTYW
        NPTYPW(ITYW)=NPTYPW(ITYW)*NIDEML
 1100 CONTINUE
C
C         testfase
C
      NTY9=0
      DO 2000 IDIFT=1,NDIFT
C
        DO 2100 ITYABT=1,NTYABT(IDIFT)
C
          TYPE=TYPEAT(IDIFT,ITYABT)
          NWORD=NPTYAT(IDIFT,ITYABT)
          WORD(2)=MOD(TYPE,10)
          WORD(1)=(TYPE-WORD(2))/10
C
          DO 2150 I=1,2
            IF (WORD(I).EQ.9) NTY9=NTY9+NWORD
 2150     CONTINUE
C
 2100   CONTINUE
C
```

```
 2000 CONTINUE
C
      NTYW=NTYW+1
      NPTYPW(NTYW)=NTY9*NIDEMT
C
C
C
      MIMATY(1,1)=MINIM
      NEXTWL(1)=MINIM
      MIMATY(1,2)=MINIM+NPTYPW(1)-1
      MIMATY(1,3)=MIMATY(1,2)-MIMATY(1,1)+1
      NEXTWT(1)=(MIMATY(1,2)+MIMATY(1,1))/2
C
      DO 3000 ITYW=2,NTYW
C
        MIMATY(ITYW,1)=MIMATY(ITYW-1,2)+1
        NEXTWL(ITYW)=MIMATY(ITYW,1)
        MIMATY(ITYW,2)=MIMATY(ITYW,1)+NPTYPW(ITYW)-1
        MIMATY(ITYW,3)=MIMATY(ITYW,2)-MIMATY(ITYW,1)+1
        NEXTWT(ITYW)=(MIMATY(ITYW,2)+MIMATY(ITYW,1))/2
C
 3000 CONTINUE
C
C
      RETURN
      END
```