stichting
mathematisch
centrum

MC

LR 2.5

APRIL 1972

D. GRUNE
HANDLEIDING BIJ DE TEKSTSCHAAF

2e boerhaavestraat 49 amsterdam

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

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

1. Doel.

Het doel van het programma 'tekstschaaf' is het uitlijnen en pagineren van teksten die in een natuurlijke taal geschreven zijn (d.w.z. niet van programmateksten). Van deze tekst wordt verwacht dat hij uit hoofdstukken bestaat welke op hun beurt uit alinea's bestaan; deze structuur moet op de invoerband door speciale symbolen aangegeven worden. Het is mogelijk de uitlijnende werking van de tekstschaaf voor bepaalde gedeelten van de invoerband uit te schakelen; de indeling in pagina's wordt echter altijd aangebracht. Het programma breekt geen woorden af.

Deze handleiding is door de tekstschaaf uitgelijnd.

2. Inleiding.

2.1. De invoer bestaat uit een reeks hoofdstukken die elk uit een reeks alinea's bestaan. De bladzijde-indeling binnen de hoofdstukken wordt door het programma geregeld; elk hoofdstuk begint op een nieuwe bladzijde. Bovenaan elke bladzijde komt een kopje, bestaande uit links een gegeven tekst en rechts een hoofdstuknummer en een bladzijdenummer; dit bladzijdenummer loopt automatisch op, het hoofdstuknummer moet aan het begin van elk hoofdstuk gespecificeerd worden.

2.2. Alinea's bestaan in twee soorten; de ene soort bestaat uit een reeks woorden, de tweede wordt beschouwd als een ondeelbare, niet-uitlijnbare eenheid. Een alinea die uit woorden bestaat, wordt uitgelijnd tot een blok tekst met rechte linker en rechter kantlijn, waarbij de eerste regel over een volle breedte van een bladzijde wordt uitgelijnd terwijl bij de tweede en volgende regels eventueel een linker marge aangehouden kan worden. De uitgelijnde alinea heeft dus de volgende vorm:

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXX....

2.3. In een alinea tekst in een natuurlijke taal zijn meestal een aantal punten aan te wijzen die zich goed lenen voor het aanbrengen van de spaties die nodig zijn voor het verkrijgen van een rechter kantlijn. De meest voor de hand liggende plaatsen zijn: direkt achter een punt, komma, dubbele punt, puntkomma of vraagteken (een uitroepteken bestaat niet in de invoercode). Hier is als het ware de binding tussen de woorden van de alinea het zwakst. Voor het vinden van andere geschikte punten is in principe redekundige ontleding van de zin nodig; dit ligt echter buiten het kader van een programma als het onderhavige. Wel zien we bij het beschouwen van ontlede nederlandse en engelse zinnen dat in veel gevallen, waar een kort woord door een langer gevolgd wordt, deze twee woorden tot dezelfde eenheid behoren. De binding tussen twee zulke woorden is als het ware sterker dan tussen andere woorden.
De waarde die de gebruiker op esthetische of andere gronden hecht aan bovenstaande overwegingen kan in getalvorm aan het programma kenbaar gemaakt worden.

2.4. Het programma breekt geen woorden af. In de meeste gevallen geeft het programma door het verdelen van spaties over meerdere regels (zie 4.2.) voor een niet al te kleine regelbreedte ook zonder afbrekingen goede resultaten.

3. Invoer en resultaat.

In dit hoofdstuk wordt een beschrijving van de input gegeven in Backus-Naur-form; bij elke regel wordt vermeld wat het effect van bepaalde input op het resultaat is. Zo nu en dan wordt de precieze betekenis van een Backus-Naur-regel in de begeleidende tekst licht gewijzigd; dit is gedaan om de formele beschrijving niet nodeloos ingewikkeld te maken.

3.1. <input> ::= <readvar block> <chapters>
Het 'readvar block', waarvan een beschrijving te vinden is in LR2.3., moet de volgende vorm hebben:

begin kopje(" <text> ") end .

De meegegeven 'text' wordt naar rechts tot een lengte van 80 symbolen aangevuld met spaties; dit is de tekst die boven aan elke pagina afgedrukt wordt, voorzien van hoofdstuknummer en paginenummer. De 'text' mag geen tabs of nlcrs bevatten; alleen de eerste 80 symbolen van de tekst zijn van belang. De beschrijving die U nu leest begint dus met :

```
begin kopje("LR2.5.                                    APRIL 1972") end
```

3.2. <chapters> ::= <chapter> ; <chapters> | <chapter>
<chapter> ::= <readvar block> <alineas>
De <chapters> bestaan uit een reeks hoofdstukken gescheiden door doorbalkte puntkomma's; achter het laatste hoofdstuk hoeft geen doorbalkte puntkomma te volgen.
Aan elk hoofdstuk gaat een 'readvar block' vooraf, waarin gegevens verschaft kunnen worden over het onderhavige hoofdstuk. Voor gegevens die bij een bepaald hoofdstuk niet gespecificeerd worden, blijven de eerder verschafte gegevens van kracht, of bij het ontbreken van deze, de begingegevens geleverd door het programma. De namen die binnen het 'readvar block' beschikbaar zijn, zullen hier als gewone ALGOL-variabelen behandeld worden.

3.2.1. De volgende gegevens kunnen verschaft worden:

integer hoofdstuk,
geldt voor het gehele hoofdstuk; minimaal 0, beginwaarde 1.

integer bladzijde,
het nummer van de eerste bladzijde van het hoofdstuk, minimaal 0, beginwaarde 1.

integer laatste,

de maximale waarde van het bladzijdenummer; blijkt op de bladzijde met als bladzijdenummer 'laatste' nog een bladzijde te volgen, dan krijgt deze als nummer weer 'laatste', nu echter gevolgd door een a, de daaropvolgende krijgt 'laatste' gevolgd door een b, enz. Hiermee kunnen dus in een bestaande tekst bladzijden tussengevoegd worden. De minimumwaarde is 0, de beginwaarde 100 000.

integer breedte,
het aantal posities op een regel. Minimaal 18 (noodzakelijk om onder alle omstandigheden voldoende ruimte te hebben voor een hoofdstuknummer en een bladzijdenummer), beginwaarde 75.

integer hoogte,
het aantal regels per bladzijde. De eerste regel wordt in beslag genomen door het kopje, de drie volgende regels zijn leeg, terwijl de daarop volgende regels met tekst gevuld kunnen worden. Geen enkele bladzijde bevat meer dan 'hoogte' regels. Minimum waarde 5, beginwaarde 56.

integer alinea.
Een alinea waarvan het aantal regels na uitlijnen gelijk aan of kleiner dan 'alinea' is, wordt nooit over twee bladzijden verdeeld. De waarde van 'alinea' moet minstens 4 kleiner zijn dan die van 'hoogte', en mag niet negatief zijn; de beginwaarde is 10.

integer kantlijn,
de breedte van de linker marge, welke bij de tweede en volgende regel van een uitgelijnde alinea in acht genomen wordt. Door deze nul te kiezen wordt het effect van de linker marge teniet gedaan. De waarde van 'kantlijn' moet kleiner zijn dan die van 'breedte', en mag niet negatief zijn; beginwaarde 8.

boolean pons,
geeft aan of er van het resultaat een ponsband geproduceerd moet worden; beginwaarde true.

integer diepte,
het aantal regels in een alinea (geteld na het uitlijnen) dat door het uitlijnproces tegelijkertijd in beschouwing genomen kan worden. Als de waarde van 'diepte' te klein gekozen wordt, gaat de uitgelijnde tekst onnodig veel spaties bevatten; is 'diepte' te groot, dan duurt het uitlijnen van lange alinea's onnodig lang. Zie hiervoor 4.2.. De beginwaarde is 3, de minimumwaarde is 1; als diepte = 1 wordt de alinea regelsgewijs 'aangestampt'.

integer zwak,middel, sterk,lzwak,lmiddel,lsterk.

Met deze variabelen kan de kracht geregeld worden waarmee het uitlijnproces bepaalde woorden bij elkaar probeert te houden (zie 2.2.). De variabelen 'zwak', 'middel' en 'sterk' bepalen de kracht waarmee zwak–, middelsterk– en sterk–verbonden woorden beschermd worden tegen het tussenvoegen van spaties, de variabelen 'lzwak', 'lmiddel' en 'lsterk' bepalen de bescherming tegen het tussenvoegen van regelovergangen. Bij voorbeeld zullen bij een zeer grote waarde van 'lsterk' twee sterk–verbonden woorden nooit over twee verschillende regels verdeeld worden. De minimumwaarden zijn 1, de beginwaarden (waarmee ook deze handleiding vervaardigd is) zijn:

zwak	1	lzwak	1
middel	2	lmiddel	1
sterk	4	lsterk	1 .

integer rafels,
geeft aan hoeveel waarde gehecht wordt aan een exact rechte rechter kantlijn. Als deze waarde veel kleiner is dan die van 'zwak', 'middel' enz., verdwijnt het uitlijneffect geheel; elke regel wordt dan gevuld met zoveel mogelijk woorden zonder dat ze rechts uitsteken, en deze worden zover mogelijk naar links geschoven. De beginwaarde van 'rafels' is 100, de minimumwaarde 1.

3.2.2. Voorbeelden.

a. begin hoofdstuk:= 3; bladzijde:= 1 end
Dit 'readvar block' is gelijk aan het 'readvar block' aan het begin van het hoofdstuk dat U nu leest.

b. begin hoofdstuk:= 3; bladzijde:= laatste:= 13 end
De bladzijden worden genummerd: 3–13, 3–13a. 3–13b, enz..

c. begin pons:= false; bladzijde:= 14; laatste:= 12 end
De bladzijden worden nu genummerd: 3–12b, 3–12c, 3–12d, enz., terwijl geen ponsband wordt geproduceerd.

d. begin hoogte:= 10 000 000; kantlijn:= 0; diepte:= rafels:= 1;
zwak:= middel:= sterk:= lzwak:= lmiddel:= lsterk:= 1000
end
De tekst wordt zonder bladzijde–indeling, zonder kantlijn en zonder rechte rechter kantlijn zo dicht mogelijk ingepakt.

3.3. <alineas> ::= <alinea> <alineas> | <alinea>
<alinea> ::= <layout option> <quoted block> ¶ |
<layout option> <words> ¶

De 'layout option' kan bestaan uit spaties, tabs en terug–wagen–nieuwe–regels; deze layout wordt gecopieerd en bepaalt het beginpunt van de uitgelijnde alinea. Als de alinea op een nieuwe bladzijde begint, worden eventueel eraan voorafgaande regelovergangen niet uitgevoerd.

Een 'quoted block' begint en eindigt met een doorbalkt vraagteken; tussen deze doorbalkte vraagteken mogen alle symbolen voorkomen met uitzondering van doorbalkte vraagteken. Deze laatste mogen evenwel wel paarsgewijs voorkomen zonder enig ander symbol ertussen; ze stellen dan in de uitvoer een enkel doorbalkt vraagteken voor. Een 'quoted block' wordt van zijn eerste en laatste doorbalkte vraagteken ontdaan en vervolgens zonder uitlijnen gecopieerd; ook de linker marge vanaf de tweede regel, die bij uitgelijnde alinea's van toepassing is, wordt hierbij niet toegevoegd.

3.4. <words> ::= <word> | <word> <separator> <words>

Een 'separator' bestaat uit spaties, tabs en/of regelovergangen; de precieze opbouw doet niet terzake, wel moet minstens 1 spatie, tab of regelovergang aanwezig zijn.

3.5. <word> ::= <normal word> | <quoted word> <mark option>

Een 'normal word' kan alle ponsbare symbolen bevatten met uitzondering van spaties, tabs, regelovergangen en doorbalkte aanhalingstekens (‽, aangezien deze dienen om een alinea af te sluiten); verder mag een 'normal word' niet met een doorbalkt vraagteken (⸘) beginnen. Een 'normal word' wordt in zijn geheel gecopieerd en zijn plaats in de alinea wordt door het uitlijnproces bepaald.
Een 'quoted word' begint en eindigt net als een 'quoted block' met een doorbalkt vraagteken. In een 'quoted word' mogen echter geen tabs en regelovergangen voorkomen. Dubbele doorbalkte vraagtekens worden ook hier als enkele doorbalkte vraagtekens verwerkt. Een 'quoted word' wordt van zijn eerste en laatste doorbalkte vraagteken ontdaan, en zijn plaats in de alinea wordt door het uitlijnproces bepaald. (de ongewisheid van de plaats maakt het onmogelijk in een 'quoted word' tabs en regelovergangen toe te laten).
Een 'mark option' is een punt, komma, dubbele punt, puntkomma of vraagteken, of wordt weggelaten. De 'mark option' vormt samen met het 'quoted word' een onverbrekelijk geheel. Naast ⸘n:= n + 1;⸘ is dus ook toegestaan ⸘n:= n + 1⸘; met dezelfde betekenis.

4. Werking van het programma.

Het programma werkt alineagewijs; voor elke alinea is er een inleesfase, een uitlijnfase en een uitvoerfase.

4.1. De inleesfase.

De woorden in de alinea worden in een dubbelgelinkte lijst opgeslagen, 1 symbool per integer. In de schakels van de lijst is ruimte voor latere uitlijngegevens. Wanneer de hele alinea ingelezen is, wordt in de schakels de sterkte van de binding van de twee bij de schakel behorende woorden genoteerd, volgens de criteria uit hoofdstuk 2.. De inleesfase is in CDL geschreven; de structuur van de toegestane invoer en die van de schakels in de lijst blijken duidelijk uit de CDL-tekst (zie 6.2.).
Voor een beschrijving van CDL zie MR127/71.

4.2. Het uitlijnen.

De door het invoerproces gemaakte lijst van woorden wordt doorgegeven aan een algoritme, die als volgt de regelindeling bepaalt. De eerste regel wordt (denkbeeldig) met zoveel mogelijk woorden gevuld, hiervan wordt de mate van uitlijnbaarheid bepaald, en de rest van de lijst wordt als een nieuwe alinea beschouwd waarvan door hetzelfde proces recursief de uitlijnbaarheid bepaald wordt; deze twee uitlijnbaarheden worden bij elkaar opgeteld en leveren de uitlijnbaarheid van deze alinea bij deze regelindeling. Nu wordt het laatste woord van de eerste regel verwijderd van deze regel en toegevoegd aan de rest van de alinea; van regel en alinea wordt bij deze nieuwe regelindeling opnieuw de uitlijnbaarheid bepaald; het is duidelijk dat de uitlijnbaarheid van de regel achteruitgegaan is, de rest van de alinea kan echter zoveel beter uitlijnbaar zijn geworden dat het totaalbeeld toch gunstiger wordt. Pas nadat op deze manier het optimale aantal woorden voor de eerste regel bepaald is, wordt de verdeling van de spaties binnen de regel bepaald en wordt de indeling definitief. Hierna wordt het proces iteratief herhaald voor de rest van de alinea.
De tijd benodigd voor de bovenstaande algoritme is (ongeveer) evenredig met $2 \uparrow r$, waarin r het aantal regels in de uitgelijnde alinea is. Dit betekent dat het uitlijnen van een alinea van 15 regels 1000 maal zoveel tijd kost als van een alinea van 5 regels. Dit is catastrofaal; bovendien draagt het vooruitkijken over meer dan enkele regels niet meer bij tot het esthetische effect. De recursiediepte in de bovenstaande algoritme is dan ook begrensd door de variabele 'diepte'. Voor een alinea van r regels is de benodigde tijd dan:

als $r > diepte$: $(r - diepte + 1) \times 2 \uparrow diepte$

als $r \leq diepte$: $2 \uparrow r$.

Met diepte = 6 is de verhouding voor twee alinea's van 15 en 5 regels 20:1 .

In het bovenstaande wordt gebruik gemaakt van een numerieke waarde voor de uitlijnbaarheid van een regel. Deze wordt als volgt uit de woorden en de regelbreedte bepaald. Uit de lengte van de woorden en de regelbreedte volgt het aantal spaties dat ingevuld moet worden, N, en het aantal groepen spaties, n. Dus is:

N = sum(i, 1, n, s[i])

als s[i] de lengte van de i–de groep spaties is. We definieren nu de uitlijnbaarheid als:

– sum(i, 1, n, k[i] × s[i] ↑ 2),

waarin k[i] de bindingskracht is tussen twee woorden. Vele andere definities zijn denkbaar; de uitlijnbaarheid wordt snel kleiner als een van de spatiegroepen groot wordt. Als we de waarden van s niet tot gehele getallen beperken, heeft de uitlijnbaarheid een maximumwaarde als:

s[j] = (N/sum(i, 1, n, 1/k[i]))/k[j] ;

deze maximumwaarde is:

– N ↑ 2/sum(i, 1, n, 1/k[i]).

Deze waarde wordt bij het bepalen van de regelindeling als schatting van de uitlijnbaarheid gebruikt. Voor het aanbrengen van het juiste aantal spaties is echter een oplossing met gehele getallen nodig. Deze wordt gevonden door van een geschatte spatie–indeling uit te gaan, en hierin net zolang woorden en woordgroepen een (1) positie te verschuiven tot op geen enkele manier een verbetering verkregen kan worden.
De uitlijnfase is geschreven in ALGOL 60.

4.3. De uitvoerfase.

De precieze lengte van de alinea is nu bekend. Als er niet genoeg ruimte op de lopende bladzijde is, wordt onderzocht of de alinea zo kort is dat hij niet over twee bladzijden verdeeld mag worden; als dat het geval is, wordt een nieuwe bladzijde begonnen en worden eventuele regelovergangen uit de 'layout option' weggegooid. Vervolgens wordt de 'layout option' uitgevoerd, gevolgd door de tekst van de alinea; indien nodig wordt ergens in de alinea op een nieuwe bladzijde overgegaan.
De uitvoerfase is in CDL geschreven.

5. Foutmeldingen.

Er bestaan vier bronnen van foutmeldingen.

5.1. Tijdens de verwerking van 'readvar blocks' kunnen foutmeldingen optreden; deze hebben de vorm:

error in readvar: <getal>

Voor de betekenis van de getallen zie LR2.3..

5.2. Ten gevolge van 'readvar blocks' kunnen variabelen niet-toegestane waarden krijgen (zie 3.2.); de hierbij optredende foutmeldingen spreken voor zichzelf. Fouten van dit type geven aanleiding tot beeindiging van het programma.

5.3. Bij het lezen van een alinea kan blijken dat hij niet aan de eisen voldoet; de hierbij optredende foutmeldingen spreken voor zichzelf. De fout wordt naar redelijkheid hersteld en het programma gaat verder.

5.4.1. De foutmelding 'einde invoer'. Het programma wordt beeindigd.

5.4.2. De foutmelding 'alinea te lang'. Het programma wordt beeindigd. In de huidige versie mag een alinea ongeveer 20000 symbolen bevatten.

6. Programmateksten.

6.1. De ALGOL 60-tekst.

```
begin  comment  cdl     tekst-schaaf, 20 - 3 -1972;

  int proc resym; resym:= resymbol;

procedure out(symbol); value symbol; integer symbol;
begin   prsym(symbol);   if punch then pusym(symbol) end out;

procedure page separator;
begin
  if punch then
  begin stopcode; puhep(127); runout; runout end;
  newpage
end page separator;

procedure distribute(number of spaces) over:(number) places
  in:(p) according to:(alpha);
value number of spaces,number; integer number of spaces,number;
  integer array p; array alpha;
begin   comment
  this procedure minimizes

f(p)= sum(i, 1, number, alpha[i] × p[i] ↑ 2)

where sum(i, 1, number, p[i])= number of spaces.
although this problem for real p has a unique solution

p[j]= 1/alpha[j] ×
        (number of spaces/sum(i, 1, number, 1/alpha[i])),

several optimal solutions may exist for integer p.
one solution is then drawn from among the correct ones.
the function f(p) is never calculated,
the effect of raising p[j] by one and
lowering p[i] by one is considered instead,
starting from an initial guess for p.
;

integer i,j,count,number2; real x,y;
array remove one from p, add one to p[1:number];

for i:= number step -1 until 1 do
begin p[i]:= number of spaces / i;
  number of spaces:= number of spaces - p[i]
end guess;

number2:= number × number;
for i:= 1 step 1 until number do
```

```
begin   x:= alpha[i]; y:= p[i] × x × 2;
  remove one from p[i]:= x - y;
  add     one to   p[i]:= x + y
end;

for count:= 0, count + 1 while count ≠ number2 do
begin   if count= 0 then
  begin   i:= random × number + .5;
  j:= random × number + .5
end initial point else
begin   i:= i + 1;
  if i > number then
  begin   i:= 1; j:= j + 1;
    if j > number then j:= 1
  end i overflow
end circular step up;

if i ≠ j then
begin   if - remove one from p[i] > add one to p[j] then
  begin   comment the deal is profitable;
    p[i]:= p[i] - 1;
    x:= alpha[i]; y:= p[i] × x × 2;
    remove one from p[i]:= x - y;
    add     one to   p[i]:= x + y;
    p[j]:= p[j] + 1;
    x:= alpha[j]; y:= p[j] × x × 2;
    remove one from p[j]:= x - y;
    add     one to   p[j]:= x + y;
    count:= -1
  end deal
  end i ≠ j
  end count
end distribute;

procedure readvar(varspec,error); procedure varspec,error;
begin comment

  Voor de programmatekst van de procedure 'readvar' zie LR 2.3.;

end readvar;


procedure read string(txt);   integer array txt;
begin   integer count;

  procedure title(pr);   pr(≠kopje≠, to txt);

  procedure to txt(sym);   integer sym;
  begin
    test(sym ≠ tab sbl, ≠kopje bevat tab≠);
    test(sym ≠ nlcr sbl, ≠kopje bevat nlcr≠);
    count:= count + 1;
    if count > 81 then else txt[count]:= sym
```

```
    end to txt;

    count:= 0;    readvar(title, errorm);
    txt[if count > 80 then 81 else count + 1]:= 255
  end read string;

  proc error(s); string s;
  begin integer sp; sp:= print pos;
    nlcr; nlcr; printtext(«fout: »);
    printtext(s); nlcr; nlcr;
    space(sp)
  end error;

  procedure errorm(er);   value er;   integer er;
  begin nlcr; printtext(«error in readvar:»);
    absfixt(3,0,er);   nlcr;
    for er:= 1 step 1 until 100 do prsym(resym);
    exit
  end errorm;

  integer array title[1:81];

  integer line width, lines per page, paragraph length,
    margin width, text width, depth,
    chapter number, number current page, last page number;
  boolean punch;
  real weight of rh margin;
  array weight of space, weight of line feed[1:3];

  procedure facts(pr);   procedure pr;
  begin   pr(«pons», punch);
    pr(«breedte», line width);
    pr(«hoogte», lines per page);
    pr(«alinea», paragraph length);
    pr(«kantlijn», margin width);
    pr(«hoofdstuk», chapter number);
    pr(«bladzijde», number current page);
    pr(«laatste», last page number);
    pr(«diepte», depth);
  end facts;

  procedure taste(pr); procedure pr;
  begin   pr(«zwak», weight of space[weak ind]);
    pr(«lzwak», weight of line feed[weak ind]);
    pr(«middel», weight of space[medium ind]);
    pr(«lmiddel», weight of line feed[medium ind]);
    pr(«sterk», weight of space[strong ind]);
    pr(«lsterk», weight of line feed[strong ind]);
    pr(«rafels», weight of rh margin)
  end taste;

  procedure var(pr);   procedure pr;
  begin facts(pr);  taste(pr)  end var;
```

```
boolean error free;

procedure check(st, var); string st; integer var;
if var < 0 then
begin  nlcr; printtext(‹variabele ' ›); printtext(st);
  printtext(‹' is niet positief:›); fixt(8,0,var);
  error free:= false
end check;

procedure test(cond, mess); value cond; bool cond; string mess;
if ¬cond then
begin nlcr; printtext(mess); error free:= false end test;

integer tmax;

integer undl, vertb, space sbl, tab sbl, nlcr sbl,
  weak ind, medium ind, strong ind, new line ind, end of paragraph;

space sbl:= 93; tab sbl:= 118; nlcr sbl:= 119;
undl:= 126; vert b:= 127;
weak ind:= 3; medium ind:= 2; strong ind:= 1; new line ind:= -1;
end of paragraph:= -2;
read string(title);

  punch:= true;   line width:= 75;   lines per page:= 56;   depth:= 3;
  paragraph length:= 10;   margin width:= 8;   chapter number:= 1;
  number current page:= 1;   last page number:= 100 000;
  weight of space[strong ind]:= 4;
  weight of space[medium ind]:= 2;
  weight of space[weak ind]:= 1;
  weight of line feed[strong ind]:= 1;
  weight of line feed[medium ind]:= 1;
  weight of line feed[weak ind]:= 1;
  weight of rh margin:= 100;

restart:
  error free:= true;
  readvar(var, errorm);   taste(check);
  tmax:= available - 2000;  text width:= line width - margin width;
  test(line width > 18, ‹breedte niet groter dan 18›);
  test(lines per page > 4, ‹hoogte kleiner dan 5›);
  test(paragraph length < lines per page - 3, ‹alinea te groot›);
  test(paragraph length ≥ 0, ‹alinea negatief›);
  test(margin width ≥ 0, ‹kantijn negatief›);
  test(margin width ≤ line width, ‹kantlijn te groot›);
  test(chapter number ≥ 0, ‹hoofdstuk negatief›);
  test(number current page ≥ 0, ‹bladzijde negatief›);
  test(last page number ≥ 0, ‹nummer laatste bladzijde negatief›);
  test(depth > 0, ‹diepte kleiner dan 1›);
  test(tmax > 100, ‹geen ruimte voor tekstarray›);
  test (error free, ‹programma beeindigd wegens bovenvermelde fouten›);
  if ¬error free then exit;
```

```
begin

  procedure outtxt;
  begin  integer word length, pointer, ind, count, sym;  real limit;
  procedure dump(n);  value n;  integer n;
  begin nlcr;
    for count:= 1 step 1 until 24 do printtext(‹dump  ›);
    nlcr;  printtext(‹error›);  absfixt(3,0,n);
    count:= 10;
    for ind:= 1 step 1 until tmax do
    begin count:= count + 1;
      if count = 6 then space(5) else
      if count = 11 then
      begin nlcr;  absfixt(5, 0, ind);  prsym(66);
        absfixt(5, 0, ind + 9);  space(5);  count:= 1
      end;
      space(2);  fixt(8, 0, txt[ind])
    end;
    goto end
  end dump;

  nlcr;
  for count:= 1 step 1 until 18 do printtext(‹outtxt  ›);
  limit:= -2; ind:= 0;
  for pointer:= limit + 1 while ind ≠ end of paragraph do
  begin  if pointer + 2 > tmax then dump(4);
    word length:= txt[pointer + 2];
    if word length < 0 then dump(2);
    nlcr;  absfixt(4, 0, word length);
    nlcr;  limit:= word length + pointer + 2;
    if limit + 2 > tmax then dump(3);
    for count:= pointer + 3 step 1 until limit do
    begin  sym:= txt[count];
      if sym < 0 ∨ sym > 511 then dump(1);
      if sym>255 then begin prsym(127); sym:= sym - 256 end;
      if sym>127 then begin prsym(126); sym:= sym - 128 end;
      prsym(sym)
    end sym;
    nlcr; absfixt(4,0,txt[limit + 1]);
    ind:= txt[limit + 2];
    nlcr; fixt(4,0,ind)
  end pointer;
end:  nlcr
end outtxt;

real procedure align(allowed line width, begin pointer, test);
value allowed line width, begin pointer, test;
integer allowed line width, begin pointer, test;
```

```
begin   integer pointer, line counter;
  real sigma, tension 0, tension 1;

  procedure step forwards;
  begin   integer length next word;
    sigma:= sigma + 1/weight of space[txt[pointer + 1]];
    length next word:= txt[pointer + 2];
    allowed line width:= allowed line width - length next word - 1;
    pointer:= pointer + length next word + 3
  end step forwards;

  procedure step backwards;
  begin   integer length present word;
    length present word:= txt[pointer];
    allowed line width:= allowed line width + length present word + 1;
    pointer:= pointer - length present word - 3;
    sigma:= sigma - 1/weight of space[txt[pointer + 1]]
  end step backwards;

  if test > depth then
  begin align:= 0;   goto out  end;

  if test = 0 then line counter:= 0;
  goto first entry;

reentry:
  begin pointer:= pointer;   allowed line width:= text width;
  line counter:= line counter + 1;

first entry:
  if allowed line width ≤ 0 then error(⟨regel springt te ver in⟩);
  pointer:= begin pointer;   sigma:= 1/weight of rh margin;
  allowed line width:= allowed line width - txt[pointer];
  if txt[pointer + 1] = end of paragraph then
  begin   align:= if test = 0 then line counter + 1 else 0;
    goto out
  end end of paragraph;

  if allowed line width ≤ 0 then
  begin   step forwards;
    if test = 0 then
    begin  txt[begin pointer + 1]:= new line ind;  goto reentry  end;
    goto out
end one word on line, is either too long or fits exactly;

step forwards;
if allowed line width < 0 then
begin   if test = 0 then
  begin  txt[begin pointer + 1]:= new line ind;  goto reentry  end;
  align:= align(text width, pointer, test + 1) +
    (allowed line width + txt[pointer] + 1) ↑ 2 × weight of rh margin +
    weight of line feed[txt[begin pointer + 1]];
```

```
      goto out
    end one word on line, is too short;

  fill line:
    if txt[pointer + 1] = end of paragraph then
    begin   if test ≠ 0 then align:= 0 else
      begin
        for pointer:= begin pointer, pointer + txt[pointer + 2] + 3
          while txt[pointer + 1] ≠ end of paragraph do
         txt[pointer + 1]:= 0;
        align:= line counter + 1
      end insert single spaces in last line;
      goto out
    end last line of paragraph;

    step forwards;
    if allowed line width ≥ 0 then goto fill line;

    tension 0:= align(text width, pointer, test + 1);
    step backwards;
    tension 0:= tension 0 + allowed line width × allowed line width / sigma
          + weight of line feed[txt[pointer + 1]];

find minimum tension:
    tension 1:= align(text width, pointer, test + 1);
    step backwards;
    tension 1:= tension 1 + allowed line width×allowed line width/sigma
          + weight of line feed[txt[pointer + 1]];
    if tension 1 < tension 0 then
    begin  tension 0:= tension 1;
      if begin pointer≠pointer then goto find minimum tension
    end;
    step forwards;

    if test ≠ 0 then
    begin  align:= tension 0;  goto out  end;

    fill out(begin pointer, pointer, allowed line width);
    pointer:= pointer + txt[pointer + 2] + 3;
    goto reentry;

out:
  end align;
```

```
procedure fill out(first pointer, last pointer, number of spaces);
value first pointer, last pointer, number of spaces;
integer first pointer, last pointer, number of spaces;
if first pointer = last pointer
then txt[last pointer + 1]:= new line ind else
begin   integer number, pointer, i;
number:= 1;
for pointer:= first pointer, txt[pointer + 2] + pointer + 3
  while pointer ≠ last pointer do number:= number + 1;

begin   real array w[1:number];
  integer array s[1:number];

  i:= 0;
  for pointer:= first pointer, txt[pointer + 2] + pointer + 3
    while pointer ≠ last pointer do
  begin   i:= i + 1;
    w[i]:= weight of space[txt[pointer + 1]]
  end w;
  w[number]:= weight of rh margin;

  distribute(number of spaces, number, s, w);

    i:= 0;
    for pointer:= first pointer, txt[pointer + 2] + pointer + 3
      while pointer ≠ last pointer do
    begin   i:= i + 1;
      txt[pointer + 1]:= s[i]
    end set spaces;

    txt[last pointer + 1]:= new line ind
  end w, s
end fill out;


comment

  Hier volgt de ALGOL 60–vertaling van het CDL–gedeelte van de
  tekstschaaf, ontdaan van de eerste begin en de laatste end.
  Zie hiervoor 6.2.

  ;
end of cdl part;

  goto restart
end
```

6.2. De CDL-tekst.

```
'comment'
  tekstschaaf, beschreven in cdl, 20-3-1972.


'external' 'action' error, out, exit, page separator.

'external' 'pointer' tmax, resym.

'external' 'pointer' margin width, line width,
        number current page, chapter number, lines per page,
        paragraph length, last page number.

'external' 'list' title.

'macro' 'action'
        make = '1':='2', add = '1':='1'+'2', sub = '1':= '1' - '2',
        round = '1':='1'"/"'2'×'2', get='3':='1'['2'], put='1'['2']:='3',
        div rem = '3':= '1'"/"'2';'4':='1'-'2'×'3',
        set = '1':= '"true"', clear = '1':= '"false"'.

'macro' 'action' align txt = '1':= align(line width - '2','  '3' ',' 0).

'macro' 'predicate' equal = '1'='2', greater = '1' > '2',
        was layout = '1' = nlcr symbol ∨ '1' = tab symbol
                ∨ '1' = space symbol,
        was punctuation mark =
        '1'=point ∨ '1'=comma ∨ '1'=semicolon ∨ '1'=colon
                ∨ '1'=question mark.

'macro' 'flag' omega = '"false"'.

'macro' 'action'
        to txt = txt pnt:= txt pnt + 1';' txt[txt pnt]:= '1',
        fetch  = txt pnt:= txt pnt + 1';' '1':= txt[txt pnt].

'macro' 'pointer'
        comma          =  87, point          =  88, colon           =  90,
        semicolon      =  91, space symbol   =  93, tab symbol      = 118,
        nlcr symbol    = 119, close symbol   = 347, paragraph symbol= 377,
        quote symbol   = 378, marked quote   =1402, minus symbol    =  65,
        question mark  = 122,
        dummy char     = 255, undl symbol    = 126, bar symbol      = 127,
        end of file symbol = -4096.

'macro' 'pointer'
        strong ind = 1, medium ind = 2, weak ind = 3, unknown = 0,
        new line ind = -1, end of par ind = -2.

'macro' 'pointer' left margin = 16.
```

```
'macro' 'pointer'
        last of previous word = txt[link pnt - 1],
        prev of link = txt[link pnt],
        type of link = txt[link pnt + 1],
        next of link = txt[link pnt + 2],
        wp   of link =     link pnt + 2.

'pointer' stock, symbol, ahead, line cnt, position, line number.

'action'init headline, title part, space part, top of form, heading,
        update headline, output headline, store in headline,
        store number in headline, store page number.

'action'charin, init nxt, nxt, nxt block symbol.

'action'text, layout option, paragraph, paragraph proper, layout,
        quoted block tail, punctuation option.

'action'store block symbol, store, put link, unlink, reset link pnt,
        advance link pnt.

'action'init paragraph, close paragraph, perhaps top of form,
        evaluate link, set indicators.

'action'out block, line, line out, lines, spaces, outsym, process outpos,
        out real sym.

'pointer' outpos.
                    'flag' line empty.

'list' txt [ 1 : tmax].

'pointer' txt pnt, link pnt.


[ definitie invoer ]
text:   init headline, heading, init nxt,
        rep:   (layout option,
                        (is symbol + close symbol, page separator;
                         paragraph, :rep)
               ).

layout option:
        make + line cnt + 0, make + position + 0,
        rep:   (is symbol + nlcr symbol, make + position + 0,
                add + line cnt + 1, :rep;
                is symbol + tab symbol, add + position + 9,
                round + position + 8, :rep;
                is symbol + space symbol, add + position + 1, :rep;
               ).

paragraph:
        paragraph proper, (is symbol + paragraph symbol; error +
```

```
        'op alinea volgt geen doorbalkt aanhalingsteken').

paragraph proper – number of nlcrs:
        make + number of nlcrs + 0, init paragraph,
        read paragraph + number of nlcrs,
                close paragraph + number of nlcrs;
        error + 'alinea ontbreekt'.

read paragraph + number of nlcrs – type:
        quoted start + type,
                (equal + type + 1, add + number of nlcrs + 1,
                        store + nlcr symbol,
                        quoted block tail + number of nlcrs;
                 equal + type + 3;
                 separator, (word list; unlink);
                );
        word list.

quoted start + type:
        make + type + 0, is block symbol + quote symbol,
        rep:    (is block symbol + nlcr symbol, make + type + 1;
                 is block symbol + tab symbol, make + type + 1,
                        store + tab symbol, :rep;
                 is block symbol + quote symbol, add + type + 2;
                 store block symbol + symbol, nxt block symbol, :rep).

quoted block tail + nlcr count:
        rep:    (is block symbol + quote symbol;
                 is block symbol + nlcr symbol, add + nlcr count + 1,
                        store + nlcr symbol, :rep;
                 store block symbol + symbol, nxt block symbol, :rep).

separator:
        was layout + symbol, put link + unknown, layout.

layout: rep: (was layout + symbol, nxt, :rep; ).

word list:
        word,
        rep:    (separator, (word, :rep; unlink);  ).

word:   quoted word, punctuation option;
        word symbol, store + symbol, nxt,
        rep:    (word symbol, store + symbol, nxt, :rep; ).

punctuation option:
        was punctuation mark + symbol, store + symbol, nxt; .

quoted word – type:
        quoted start + type,
       (equal + type + 2;
        error + 'quoted word bevat tab of nlcr', equal + type + 1,
                quoted block tail + type).
```

```
word symbol:
        word delimiter, omega; .

word delimiter:
        was layout + symbol; equal + symbol + paragraph symbol.
```

```
[ invoer mechanisme ]
charin + symb – undl flag – bar flag – n:
        greater + 0 + stock,
        make + undl flag + 0, make + bar flag + 0,
skip:   make + n + resym,
       (equal + n + end of file symbol, error + 'einde invoer', exit;
        greater + 0 + n, error + 'pariteitsfout', :skip;
        equal + n + undl symbol, make + undl flag + 128, :skip;
        equal + n + bar symbol, make + bar flag + 256, :skip;
        add + undl flag + bar flag,
              (greater + undl flag + 0,
                      (equal + n + nlcr symbol,
                              make + stock + nlcr symbol,
                              make + n + space symbol, :res;
                       equal + n + tab symbol, error +
                              'onderstreepte of doorbalkte tab',
                              make + stock + tab symbol,
                              make + n + space symbol, :res;
                              res:    add + n + undl flag, make + symb + n);
                make + symb + n));
        make + symb + stock, make + stock + end of file symbol.

init nxt:
        make + stock + end of file symbol, charin + symbol,
        charin + ahead.

nxt:    make + symbol + ahead, charin + ahead.

is symbol + char:
        equal + char + symbol, nxt.

nxt block symbol:
        nxt,   (equal + symbol + quote symbol,
                       (equal + ahead + quote symbol, nxt,
                        make + symbol + marked quote;      );
               ).

is block symbol + char:
        equal + char + symbol, nxt block symbol.
```

```
[ administratie invoer ]
put link + ind – x:
        make + x + txt pnt, sub + x + link pnt, sub + x + 2,
        make + next of link + x, store + x, store + ind, store + 0,
```

```
        advance link pnt.

unlink: error + 'op separator volgt geen woord',
        make + txt pnt + link pnt, sub + txt pnt + 1,
        sub + link pnt + prev of link, sub + link pnt + 3.

advance link pnt:
        add + link pnt + next of link, add + link pnt + 3.

reset link pnt: make + link pnt + 1.

store block symbol + symb:
        equal + symb + marked quote, store + quote symbol;
        store + symb.

store + symb:
        equal + txt pnt + tmax, error + 'alinea te lang', exit;
        to txt + symb.


[ behandeling tekstarray ]
init paragraph:
        make + txt  pnt + 0,
        reset link pnt,
        store + 0, store + 0, store + 0.

close paragraph + real nlcrs – x:
        put link + end of par ind, set indicators,
        reset link pnt, advance link pnt,
        align txt + x + position + link pnt,
        add + x + real nlcrs, add + x + line cnt,
        perhaps top of form + line cnt + x,
        lines + line cnt, spaces + position,
        out block.

set indicators:
        reset link pnt, advance link pnt,
        rep:   (greater + type of link + end of par ind,
                evaluate link, advance link pnt, :rep; ).

evaluate link – n:
        make + n + last of previous word, was punctuation mark + n,
        make + type of link + weak   ind;
        greater + next of link + prev of link,
        make + type of link + strong ind;
        make + type of link + medium ind.


[ administratie opschrift ]
'list' headline [ 1 : line width].

init headline – n:
        make + n + 0, put + headline + line width + dummy char,
```

```
        title part + n, space part + n.

title part + n – char:
        rep:    (add + n + 1, get + title + n + char,
                        (equal + char + dummy char, sub + n + 1;
                         put + headline + n + char, :rep)
                ).

space part + n:
        rep:    (greater + line width + n, add + n + 1,
                 put + headline + n + space symbol, :rep; ).

top of form:
        page separator, heading.

heading:
        update headline, make + line number + 0,
        add + number current page + 1, output headline.

update headline – pnt:
        make + pnt + line width, store page number + pnt,
        store in headline + pnt + minus symbol,
        store number in headline + pnt + chapter number + 10 + 0.

output headline – n – char:
        outsym + nlcr symbol, make + n + 0,
        rep:    (greater + line width + n, add + n + 1,
                 get + headline + n + char, outsym + char, :rep;
                 outsym + nlcr symbol, outsym + nlcr symbol,
                 outsym + nlcr symbol).

store in headline + pnt + sym:
        put + headline + pnt + sym, sub + pnt + 1.

store page number + pnt – n:
        greater + number current page + last page number,
        make + n + number current page, sub + n + last page number,
        store number in headline + pnt + n + 26 +9,
        store number in headline + pnt + last page number + 10 + 0;
        store number in headline + pnt + number current page + 10 + 0.

store number in headline + pnt + number + base + offset – quot – rest:
        divrem + number + base + quot + rest, add + rest + offset,
        store in headline + pnt + rest,
                (equal + quot + 0;
                 store number in headline + pnt + quot + base + offset).

perhaps top of form + caption + required – lines left:
        make + lines left + lines per page, sub + lines left + line number,
        add + lines left + 1,
       (greater + lines left + paragraph length;
        greater + lines left + required;
        make + caption + 0, top of form).
```

```
[ uitvoer ]
out block:
        reset link pnt, line,
        rep:   (link, line, :rep; ).

link:
        equal + type of link + 0, spaces + 1;
        greater + type of link + 0, spaces + 1, spaces + type of link;
        equal + type of link + new line ind,
                outsym + nlcr symbol, spaces + margin width.

line:   line out, advance link pnt.

line out – count – symb:
        make + txt pnt + wp of link, make + count + next of link,
        rep:   (greater + count + 0, fetch + symb,
                outsym + symb, sub + count + 1, :rep;  ).

lines + numb – count:
        make + count + numb,
        rep:   (greater + count + 0, outsym + nlcr symbol,
                        sub + count + 1, :rep; ).

spaces + numb – count:
        make + count + numb,
        rep:   (greater + count + 0, outsym + space symbol,
                        sub + count + 1, :rep; ).

outsym + symb:
        equal + symb + nlcr symbol, out + nlcr symbol,
                make + outpos + left margin, set + line empty,
                add + line number + 1,
               (greater + line number + lines per page, top of form; );
        line empty,
               (equal + symb + space symbol, add + outpos + 1;
                equal + symb + tab symbol,
                        add + outpos + 9, round + outpos + 8;
                clear + line empty, process outpos, out real sym + symb);
        out real sym + symb.

process outpos:
        tabs:  (greater + outpos + 7,
                        out + tab symbol, sub + outpos + 8, :tabs;
                sps:    greater + outpos + 0, out + space symbol,
                        sub + outpos + 1, :sps;
               ).

out real sym + symb – cs:
        greater + symb + 255, out + bar symbol, make + cs + symb,
                sub + cs + 256, out real sym + cs;
        greater + symb + 127, out + undl symbol, make + cs + symb,
```

```
                    sub + cs + 128, out real sym + cs;
        out + symb.

'result' text.
```

6.3. Gebruikte procedures.

De tekstschaaf gebruikt de volgende procedures uit het Milli–systeem. Voor een beschrijving van deze procedures zie LR 1.1. .

absfixt
available
exit
fixt
newpage
nlcr
print pos
printtext
prsym
puhep
prsym
random
read1
resymbol
runout
space
stopcode
stringsymbol

6.4. Het uitlijnen van de eerste vijf hoofdstukken van deze handleiding kostte 43 milli–uur; dat is twee regels per seconde, of ongeveer twintig seconden per bladzijde.