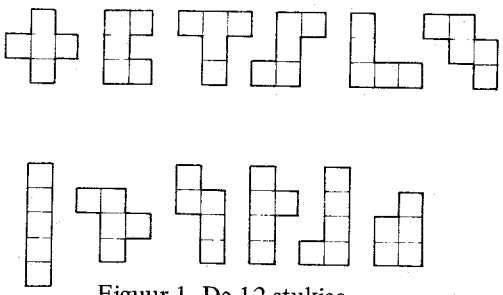
Programmeren van een pentomino puzzle

Programmeren van de pentomino puzzle

PROF. DR. N.G. DE BRUIJN [1]
Eindhoven

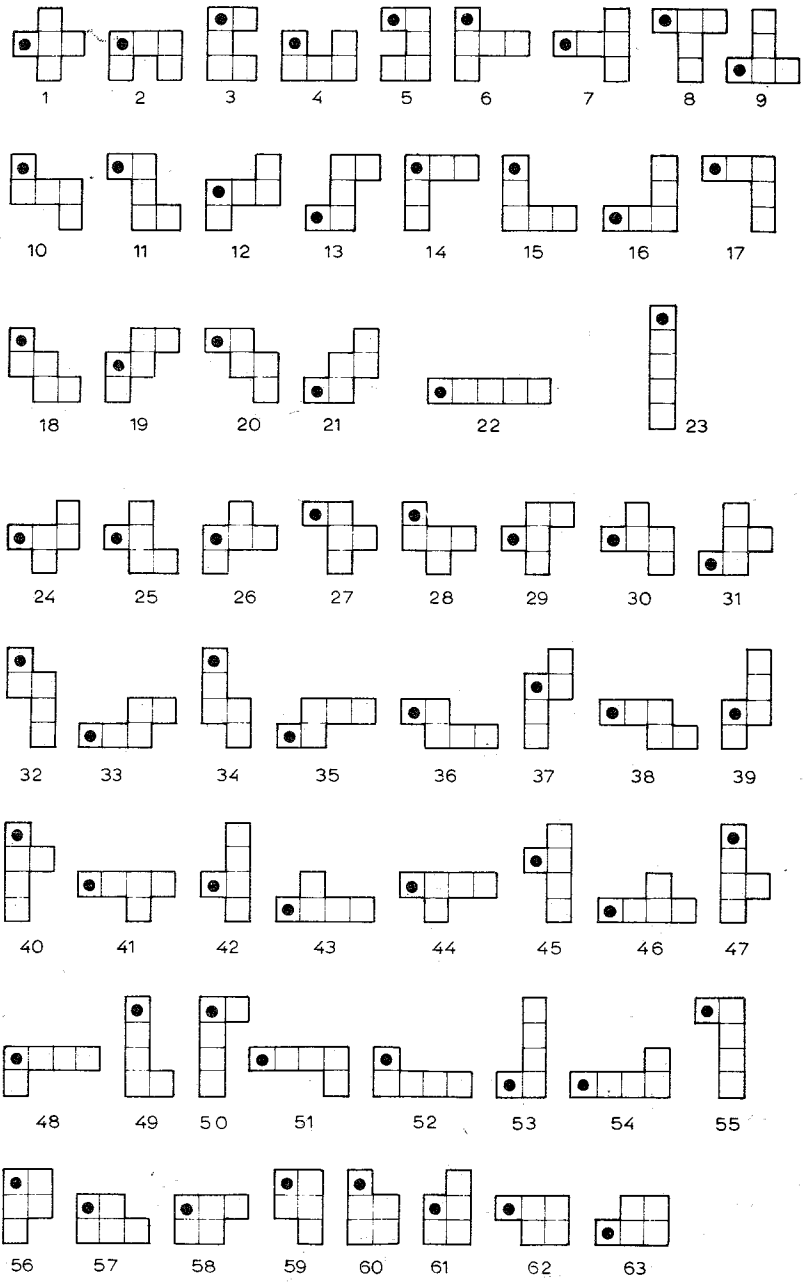
1 We nemen ons voor om met behulp van een computerprogramma alle oplossingen te vinden van de 6 x 10 pentomino [2]. Dit is een legpuzzle, waarbij gevraagd wordt om een rechthoek van 6 lengteeënheden hoog en 10 lengteeenheden breed (het *'bordje'*) te vullen met 12 gegeven *'stukjes'*, getekend in fig. 1. Elk dezer stukjes kan men opgebouwd denken uit 5 eenheidsvierkantjes door aanelkaarplakken langs gehele zijden. In feite zijn het precies alle 12 verschillende stukjes die men zo kan krijgen. Twee stukjes worden gelijk genoemd wanneer ze congruent zijn; ook gespiegelde figuren heten congruent. Dit laatste hangt samen met het feit dat de stukjes mogen worden omgeklapt: ze hebben geen duidelijke vóór of achterkant.



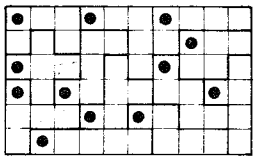
Figuur 1. De 12 stukjes.

Men voelt gemakkelijk aan, en kan met wat moeite ook wel bewijzen, dat in een oplossing een stukje alleen zó op het bord kan komen te liggen dat de eenheidsvierkantjes van het stukje samenvallen met een vijftal van de 60 eenheidsvierkantjes waarin men het bordje verdeeld kan denken. Wanneer men afziet van translaties, kunnen de stukjes, behalve het eerste (het kruis), nog op verschillende manieren op het bordje worden gelegd. Op deze manier ontstaan 63 figuurtjes die we *'plakjes'* zullen noemen, getekend in figuur 2. De in figuur 1 getekende stukjes geven aanleiding tot resp. 1, 4, 4, 4, 4, 4, 2, 8, 8, 8, 8, 8 plakjes.

2 Men kan aan elke oplossing een serie van 12 plakjes toevoegen zó dat verschillende ~~plakjes~~ oplossingen verschillende series opleveren. Dit kan op vele manieren gebeuren; we zullen ons houden aan de volgende manier, die we aan de hand van fig. 3 en fig. 4 beschrijven. In de oplossing van fig. 3 gaan we de 60 vierkantjes doorlopen, beginnende met de eerste kolom van boven naar beneden, dan de tweede kolom van boven naar beneden, enz. Iedere keer dat we voor het eerst in een nieuw plakje komen, plaatsen we een *merkteken*, en we noteren de plakjes in de volgorde waarin we ze aantroffen. Dat is gebeurd in figuur 4. We zullen zo'n serie van plakjes een *woord* noemen, de plakjes heten *letters* en de collectie van 63 plakjes heet het *alfabet*.

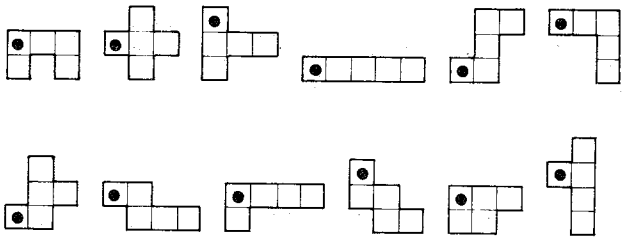


Figuur 2. De 63 plakjes.



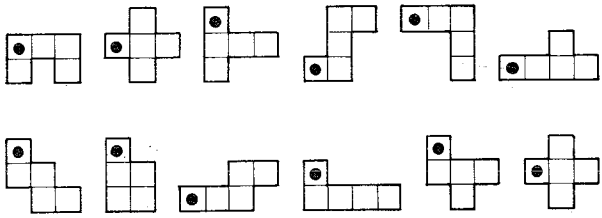
Figuur 3. Een oplossing.

Het is duidelijk dat bij elk plakje het merkteken in de meest links gelegen kolom van het plakje terechtkomt, en in die kolom op het hoogstgelegen vierkant. De ligging van het merkteken op het plakje hangt dus niet van de oplossing af; we kunnen de merktekens op de plakjes à priori aangeven. In fig. 2 is dat gebeurd.
Omgekeerd kan men uit het woord van fig. 4 de oplossing uit fig. 3 terugvinden. Men legt het eerste plakje neer met zijn merkteken op het vierkantje links-boven. Men doorloopt de vierkantjes en zoekt het eerste onbezette veld (*'eerstegat'*); men legt het tweede plakje daar met zijn merkteken op. Men doorloopt verder de vierkantjes tot wat nu weer het eerstegat is, en legt daarop het derde plakje neer, enz. Met 'doorlopen' wordt hier natuurlijk bedoeld het aftasten van kolom na kolom, elk van boven naar beneden.



Figuur 4. Het 'woord' van de oplossing van figuur 3.

Niet elke serie van 12 'letters' is een bruikbaar woord. Als we op het woord van fig. 5 het in de vorige alinea geschetste procédé toepassen, loopt het bij het vierde plakje spaak wegens overlapping. Ook op andere wijze zien we dat het woord niet met een oplossing correspondeert: het kruis komt er twee keer in voor! We zullen echter niet naar fouten achter in een woord gaan kijken, en steeds van het begin af aan het woord lezen, en letter na letter acceptabel verklaren totdat we eventueel op een onacceptabele letter stuiten. Onacceptabel kan betekenen (i) het overlapt reeds gelegde plakjes, (ii) het steekt over de rand heen, (ii) het stukje is niet meer beschikbaar omdat we het in de één of andere stand al op het bordje hebben liggen.



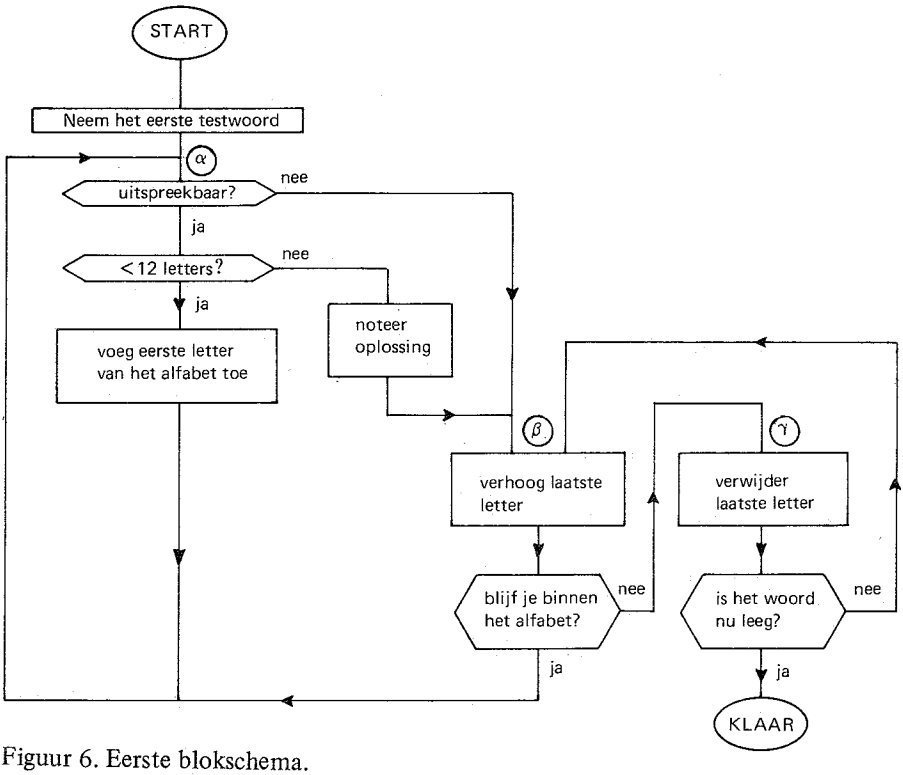
Figuur 5. Een 'onuitspreekbaar woord.'

3 Doordat we woorden van links naar rechts aftasten, kunnen we een parallel trekken met uitspreekbaarheid van een woord bij zekere uitspraakregels. Ook daarbij kunnen we woorden beoordelen door van links af te werken (wanneer tenminste de taal zó is dat een beginstuk van een uitspreekbaar woord ook uitspreekbaar is). Een woord van 12 of minder letters, met het 63-letterige alfabet van fig. 2, zullen we dus *uitspreekbaar* noemen wanneer de plakjes van dat woord achtereenvolgens op het bordje kunnen worden gelegd, volgens de regel van merkteken op eerstegat, en zonder dat enig plakje wegens één der boven genoemde regels (i), (ii), (iii) onacceptabel zou zijn.

4 We beschouwen nu een alfabetisch geordende lijst van alle woorden van 12 of minder letters. De volgorde in het alfabet is die van fig. 2. We zoeken daarin alle uitspreekbare woorden, en in het bijzonder de uitspreekbare van 12 letters. Laatstgenoemde zijn de oplossingen van ons probleem.
Er is geen beginnen aan om de *gehele* woordenlijst door een computer op uitspreekbaarheid te laten testen. Die lijst bevat nl. $63 + 63^2 + \ldots + 63^{12}$ woorden, en dat zou een snelle hedendaagse computer met het slimste programma nog wel een honderd millioen jaar kosten. De tegenwerping dat we gedurende een groot deel van die periode over véél snellere apparaten zullen beschikken is voor ons op dit ogenblik een schrale troost. Wat echter heel goed mogelijk blijkt, is om een computer alle uitspreekbare woorden te laten doorlopen, en daarvan uit die lijst alle 12-letterige af te drukken. Dat hoeft een snelle computer bij geschikte programmering tegenwoordig niet veel meer dan een half uurtje te kosten.
Om de computer in staat te stellen de uitspreekbare woorden te vinden laten we hem de *testwoorden* doorlopen. Een testwoord is een woord van $\leqslant 12$ letters dat uitspreekbaar wordt door de laatste letter weg te laten. De uitspreekbare woorden zijn dus ook testwoorden. Het lege woord wordt als uitspreekbaar beschouwd, zodat alle woorden van één letter testwoorden zijn.
Héél ruw hebben we nu als programma:
I Begin bij het eerste testwoord.
II Als het beschouwde testwoord een uitspreekbaar woord van 12 letters is, noteer dat dan als oplossing.
III Ga na of er een volgend testwoord is. Is er geen te vinden, dan is de testwoordenlijst beeindigd; is er wèl een, ga dan daarmee naar II.

5 We kijken nu wat er bij III gedaan moet worden. Als ons testwoord van

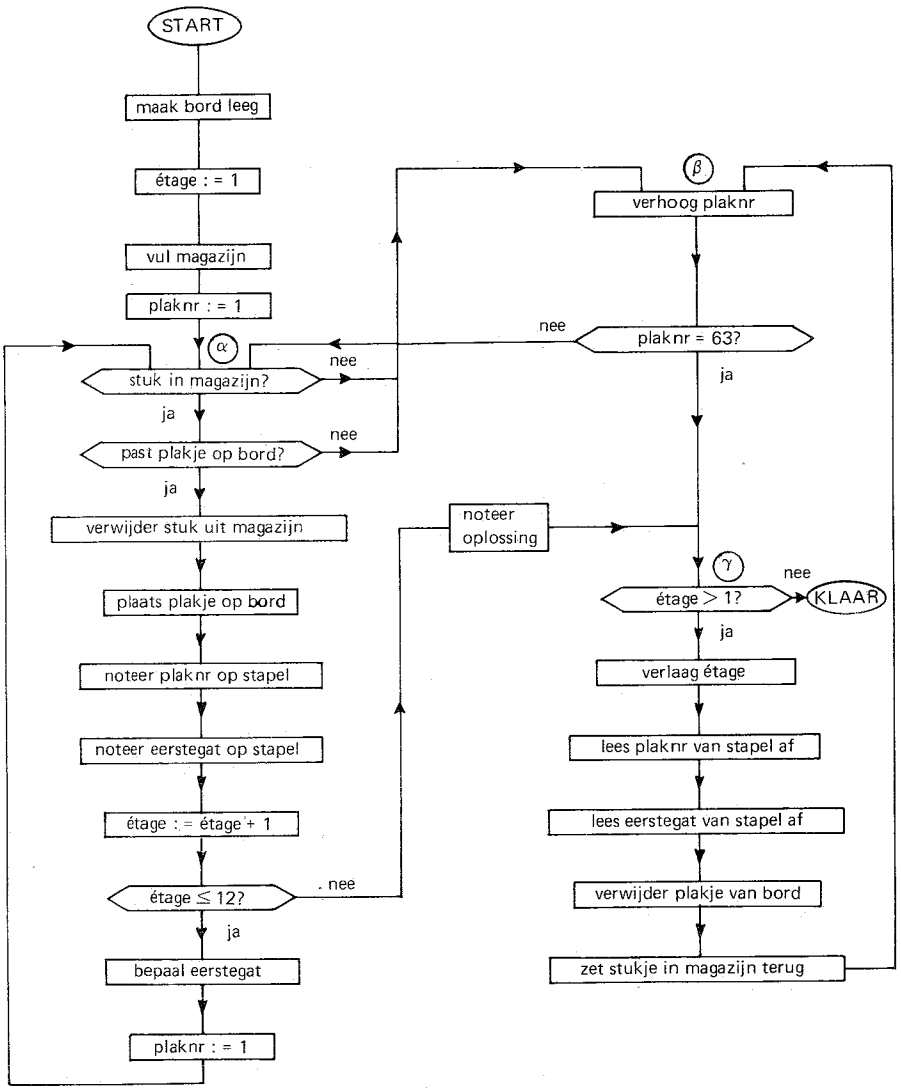
uitgang uitspreekbaar is met minder dan 12 letters, dan wordt het eerstvolgende testwoord verkregen door de eerste letter van het alfabet er achter aan te hangen. In alle andere gevallen wordt het volgende testwoord gezocht door de laatste letter te 'verhogen', d.i. te vervangen door de daarop volgende letter van het alfabet. In het geval dat de laatste letter van het testwoord tevens de laatste letter van het alfabet is, verwijderen we de laatste letter, en we vervangen de laatste letter van het nieuwe woord door de daaropvolgende letter van het alfabet. Wàs het al de laatste letter van het alfabet dan breken we verder af, enz. Zie figuur 6.
Als ons testwoord wèl uitspreekbaar is, maar precies 12 letters heeft, is er geen kans meer op uitspreekbaarheid door de laatste letter te veranderen, want het gat dat door weghalen van het laatste plakje ontstaat, heeft de oppervlakte 5, en daar past geen ander plakje in. Dit betekent dat we iets kunnen overslaan; in het blokschema van fig. 6 kunnen we desgewenst direct van de nee-uitgang van '< 12 letters? ' naar het punt γ lopen i.p.v. naar punt β.



Figuur 6. Eerste blokschema.

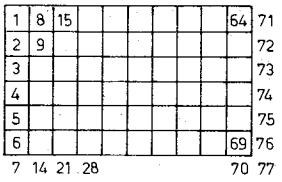
6 We willen nu aangeven hoe de vraag 'uitspreekbaar' uit het blokschema van fig. 6 wordt behandeld. Daar de vraag alleen voor testwoorden wordt gesteld is het al bekend dat het alleen nog maar om de laatste letter gaat, want na weglating van die letter is het woord uitspreekbaar. Met dit uitspreekbare stuk correspondeert

een stel op het bord gelegde plakjes (vgl. § 2). We zullen deze 'bordsituatie' niet uit dat woord hoeven op te bouwen, want de wijzigingen in dat woord betreffen alleen maar de laatste letter (toevoeging, weglating of wijziging). We zullen dus steeds de bordsituatie onthouden en bijwerken als dat te pas komt. Verder zullen we een lijst van de letters van het woord moeten bijhouden. Is k het aantal letters van het testwoord, dan moeten we de 1e letter, ..., $(k-1)^e$ letter ergens noteren, alsmede de (misschien niet acceptabele) k^e letter. De rij letters 1^e t.e.m. $(k-1)^e$ wordt de *'stapel'* genoemd. We noemen het een stapel omdat de enige operaties die we uitvoeren zijn: wegnemen van de bovenste resp. bovenop leggen van een nieuwe. De eerste letter ligt onderaan. Het getal k wordt de *etage* genoemd; het is de hoogte waarop we ons voornemen de eerstvolgende letter te plaatsen.



Figuur 7. Nader uitgewerkt blokschema.

7 Als we willen weten of het testwoord van k letters uitspreekbaar is, gaan we uit van de ons bekende bordsituatie van de eerste $k-1$ stukjes, en we kijken of het k^e plakje (met merkteken op het eerste gat) kan worden bijgeplaatst. De bordsituatie geeft aan welke van de 60 velden bezet zijn. Het eerste gat kan daaruit worden bepaald. We doen verstandig dat eerste gat niet iedere keer opnieuw te berekenen, maar het uit de voorafgaande toestand af te leiden. Met het oog daarop is het prettig om op de stapel niet alleen de geplaatste plakjes te vermelden, maar ook van elk plakje de positie die zijn merkteken op het bordje inneemt. Wanneer we dan plakjes aan het eind van het woord weghalen, is het nieuwe eerste gat direct van de stapel af te lezen.
Behalve de stapel, de étage, het nummer van de k^e letter (het *plaknummer*), en het bordje moeten we ook het *magazijn* bijhouden. Dat bestaat uit 12 geheugenplaatsen waarop aangetekend staat welke van de 12 stukjes nog beschikbaar zijn. De uitspreekbaarheid hangt nl. niet alleen af van de plaatsbaarheid van het k^e plakje, maar ook van de vraag of het betreffende stukje al eerder op het bord lag.
We werken nu het blokschema van fig. 6 nader uit tot dat van fig. 7. Op overeenkomstige plaatsen zijn in de blokschema's van fig. 6 en fig. 7 letters α, β, γ bijgeplaatst, opdat duidelijk is hoe fig. 7 uit fig. 6 is ontstaan. (Let op de opmerking gemaakt aan het slot van § 5).



Figuur 8. Nummering der velden
en randvelden.

8 We zullen nu voor de verschillende zaken coderingen gaan kiezen. In de eerste plaats voorzien we het bordje van een onderrand en een rechterrand, en we nummeren de velden als aangegeven in figuur 8. De velden 7, 14, 21, ..., 70 en 71 t.e.m. 77 worden steeds als bezet beschouwd. Dit zal blijken het voordeel te hebben dat de punten (i) en (ii) uit het slot van § 2 op geheel dezelfde wijze worden behandeld. Verder geven we bij elk plakje de vier z.g. *relatieve* posities aan. Van elk vierkantje van het plakje kan nl. de plaats op het bordje worden berekend door bij de plaats die het merkteken inneemt een getal op te tellen dat niet afhangt van de positie van het plakje. Zo zijn bijv. van plakje 21 (zie figuur 9) de relatieve posities 6,7,12,13. Wanneer men probeert dit plakje te plaatsen met merkteken op bordveld 2, dan berekent men door optelling dat nu plaats gevraagd wordt op 8,9,14,15. Die plaats is er niet, want zoals gezegd is veld 14 permanent bezet. Men ziet hoe de velden 7, 14,...., 70 zowel de beveiliging tegen overschrijding van onderrand als bovenrand verzorgen. De linkerrand van het bordje hoeft niet te worden beveiligd, want de relatieve posities zijn altijd positief.
De relatieve posities zullen worden bewaard in vier rijen elk ter lengte 63. Ze heten

relposeen, relpostwee, relposdrie, relposvier. Zo is bijv. relposdrie (21) = 12; het is de derde relatieve positie van plakje 21. (Om te voorkomen dat bordplaatsen > 77 ooit zullen worden geraadpleegd spreken we af dat voor elk plakje de relatieve posities in opklimmende volgorde staan).
Om de magazijnadministratie te kunnen voeren hebben we de rij 'stuknr' ingesteld. Zo is stuknr (12) = 4 omdat het 12^e plakje een stand van het 4^e stukje is.
De genoemde rijen worden gevuld door middel van een *getallenband* die al deze gegevens bevat. Op deze band staan achtereenvolgens de relatieve posities en stuknr van het eerste plakje (6,7,8,14,1), de overeenkomstige voor het tweede plakje (1,7,14,15,2), enz.
Het magazijn is een rij van 12 getallen; magazijn (i) = 1 betekent dat het i-de stukje in het magazijn is, magazijn (i) = 0 betekent dat het i-de stukje op het bordje ligt.

Figuur 9. Relatieve posities
bij plakjes 21 en 28.

9 Wanneer men een oplossing van onze puzzle heeft, kan men er direct 3 bijmaken, nl. door rotatie over 180°, door omklapping bijv. om de linkerrand, en daarna nog eens door een rotatie over 180°. We kunnen dus de oplossingen indelen in groepen van 4, en het is voldoende om er uit elke groep één aan te wijzen. Dat doen we door te eisen dat het centrum van het kruis links boven het centrum van het bordje komt te liggen. Dit betekent dat het merkteken van het kruis op één der velden 2,3,9,10,16,17,23,24 komt (het veld 2 kunnen we direct uitsluiten wegens het gaatje dat daarmee op veld 1 zou ontstaan). We splitsen nu onze puzzle in 7 kleinere, al naar gelang deze 'kruisplaats'. Zo wordt bij de eerste puzzle het kruis gefixeerd op de velden 3, 3+6, 3+7, 3+8, 3+14; deze velden worden dan permanent bezet gehouden. En we werken met de plakjes 2 t.e.m. 63 i.p.v. 1 t.e.m. 63.

10 We bespreken nu het in ECOL[3] geschreven programma. Terwille van de discussie hebben we elke ECOL-regel een nummer als label gegeven, en niet alleen aan de regels waarnaar werkelijk in het programma wordt verwezen.
In regel 1 t.e.m. 10 worden de diverse rijen gedeclareerd. We wijzen nog op *'plakstapel'* waarin de nummers van de op het bordje geplaatste plakjes worden bijgehouden, en *'gatstapel'* voor de posities van de merktekens van die plakjes. (We noemen dit 'gatstapel' omdat op het ogenblik van plaatsing van het plakje het merkteken terecht komt op wat op dat ogenblik het eerstegat is.)
In regels 11 t.e.m. 18 wordt gezorgd voor het inlezen van de getallenband.
In regel 19 t.e.m. 25 worden in de § 10 genoemde velden 3,9,10,16,17,23,24 in een rij 'kruisplaats' gezet. Regel 26 initialiseert het oplossingsnummer dat bij elke oplossing zal worden afgedrukt.
Regels 27,28, samen met 116 en 117, regelen de achtereenvolgende kruisposities:
In regels 41 t.e.m. 49 wordt het kruis op het bord geplaatst (doordat de getallen

6,7,8,14 in het programma gezet zijn, was het inlezen van deze vier getallen eigenlijk overbodig). Vóórdat dit kruis wordt ingevuld, wordt echter eerst het bord schoongemaakt (regels 29 t.e.m. 32) en de rand gevuld (regels 33 t.e.m. 40).
In regels 50 t.e.m. 54 wordt het magazijn gevuld.
Met regels 55 t.e.m. 58 wordt het veld 1 tot eerstegat gemaakt. (Vanuit regel 84 kan naar regel 56 worden teruggesprongen). In het algemeen zorgen regels 56,57,58 ervoor dat na plaatsing van een nieuw stukje op het bord verder wordt gezocht naar het eerstvolgende veld dat nog vrij is.
Door regel 59 wordt het eerste plakje aangewezen; doordat plakje 1 niet meer meedoet, is 2 het eerste plaknummer.
Bij regel 60 hebben we de in het blokschema met α aangeduide plaats; evenzo corresponderen regels 94 en 96 met β resp. γ.
Regels 60,61,62 vragen of het stuk in het magazijn is, en regel 63 t.e.m. 74 kijken of het plakje past; ingeval van mislukking komen we bij 94 terecht. Als het wèl lukt wordt het stukje uit het magazijn gehaald (regel 75) en het plakje op het bord gezet (76 t.e.m. 80). In 81 en 82 wordt het plakje mèt het op dat moment geldende eerstegat op de stapel genoteerd, en de étage verhoogd om klaar te zijn voor een volgend plakje. Als daardoor de 12^e étage bereikt is, is er een oplossing (d.i. 11 plakjes geplaatst) die (met vermelding van oplossingsnummer) wordt afgedrukt. Dit laatste gebeurt in regels 85 t.e.m. 93. Als bij regel 84 het antwoord bevestigend luidt, moet het nieuwe eerstegat bepaald worden en het plaknummer 2 gemaakt worden. Dit gebeurt door verwijzing naar regel 56, hetgeen ons weer naar regel 60 leidt.
Regels 94 en 95 corresponderen met de beide opdrachten uit het blokschema (fig. 7) bij het punt β.
Regel 96 correspondeert met γ. In plaats van 'KLAAR' komen we bij 116 terecht om een nieuwe positie van het kruis in te stellen. Regel 98 leest na het wegnemen van het laatste plakje af wat het eerstegat is: dat was de op 'gatstapel' onthouden positie van het merkteken van het weggenomen plakje. In regel 99 wordt het nummer van het plakje afgelezen, teneinde (via 115) bij 94 het volgende aan de beurt zijnde plakje te kunnen bepalen. Merk op dat bij 97 de étage is verlaagd, maar dat niet de moeite is genomen om eerst de vorige étage schoon achter te laten. Daar wordt immers niets meer afgelezen vóórdat er eerst weer overheen geschreven is.
De regels 100 t.e.m. 114 zijn de 'omkeringen' van 75 t.e.m. 80. Hier volgt nu het programma:

```
    START

1   RIJ (1:63) relposeen
2   RIJ (1:63) relpostwee
3   RIJ (1:63) relposdrie
4   RIJ (1:63) relposvier
5   RIJ (1:63) stuknr
6   RIJ (1:77) bezet
7   RIJ (1:12) magazijn
8   RIJ (1:11) plakstapel
9   RIJ (1:11) gatstapel
```

```
10  RIJ (1:7) kruisplaats
11  k := 1
12  relposeen (k):=LEES
13  relpostwee (k):=LEES
14  relposdrie (k):=LEES
15  relposvier (k):=LEES
16  stuknr (k):=LEES
17  k := k+ 1
18  ALS k > 63 DAN 19 ANDERS 12
19  kruisplaats (1):=3
20  kruisplaats (2):=9
21  kruisplaats (3):=10
22  kruisplaats (4):=16
23  kruisplaats (5):=17
24  kruisplaats (6):=23
25  kruisplaats (7):=24
26  oplnr := 0
27  i := 1
28  j := kruisplaats (i)
29  k := 1
30  bezet (k) := 0
31  k := k+1
32  ALS k > 70 DAN 33 ANDERS 30
33  k := 7
34  bezet (k) := 1
35  k := k+7
36  ALS k > 70 DAN 37 ANDERS 34
37  k := 71
38  bezet (k) := 1
39  k := k+1
40  ALS k > 77 DAN 41 ANDERS 38
41  bezet (j) := 1
42  x := j+6
43  bezet (x) := 1
44  x := j+7
45  bezet (x) := 1
46  x := j+8
47  bezet (x) := 1
48  x := j+14
49  bezet (x) := 1
50  k := 2
51  magazijn (k) := 1
52  k := k+1
53  ALS k > 12 DAN 54 ANDERS 51
54  etage := 1
55  eerstegat := 0
56  eerstegat := eerstegat+1
```

```
57  x := bezet (eerstegat)
58  ALS x = 0 DAN 59 ANDERS 56
59  plaknr := 2
60  x: = stuknr (plaknr)
61  y: = magazijn (x)
62  ALS y = 0 DAN 94 ANDERS 63
63  x := relposeen (plaknr)
64  bewaareen := eerstegat + x
65  ALS bezet (bewaareen) = 1 DAN 94 ANDERS 66
66  x := relpostwee (plaknr)
67  bewaartwee := eerstegat + x
68  ALS bezet (bewaartwee) = 1 DAN 94 ANDERS 69
69  x := relposdrie (plaknr)
70  bewaardrie := eerstegat + x
71  ALS bezet (bewaardrie) = 1 DAN 94 ANDERS 72
72  x := relposvier (plaknr)
73  bewaarvier := eerstegat + x
74  ALS bezet (bewaarvier) = 1 DAN 94 ANDERS 75
75  magazijn (stuknr(plaknr):= 0
76  bezet (eerstegat) := 1
77  bezet (bewaareen) := 1
78  bezet (bewaartwee) := 1
79  bezet (bewaardrie) := 1
80  bezet (bewaarvier) := 1
81  gatstapel (etage) := eerstegat
82  plakstapel (etage) := plaknr
83  etage := etage + 1
84  ALS etage < 12 DAN 56 ANDERS 85
85  NR
86  oplnr := oplnr + 1
87  SCHRIJF (4,0) := oplnr
88  TEKST := ":"
89  k := 1
90  x := plakstapel (k)
91  SCHRIJF (2,0) := x
92  k := k+1
93  ALS k > 11 DAN 96 ANDERS 90
94  plaknr := plaknr + 1
95  ALS plaknr ⩽ 63 DAN 60 ANDERS 96
96  ALS etage > 1 DAN 97 ANDERS 116
97  etage := etage – 1
98  eerstegat := gatstapel (etage)
99  plaknr := plakstapel (etage)
100 bezet (eerstegat) := 0
101 x := relposeen (plaknr)
102 y := eerstegat + x
103 bezet (y) := 0
```

```
104  x := relpostwee (plaknr)
105  y := eerstegat + x
106  bezet (y) := 0
107  x := relposdrie (plaknr)
108  y := eerstegat + x
109  bezet (y) := 0
110  x := relposvier (plaknr)
111  y := eerstegat + x
112  bezet (y) := 0
113  x := stuknr (plaknr)
114  magazijn (x) := 1
115  NAAR 94
116  i := i+1
117  ALS i > 7 DAN 118 ANDERS 28
118  TEKST := "klaar"
119  KLAAR
```

GETALLENBAND (bevat 63 x 5 getallen)

```
7,8,14,1,       1,7,14,15,2,    1,2,7,9,2,      1,8,14,15,2,    2,7,8,9,2,      1,2,8,15,3,
13,14,15,3,     7,8,9,14,3,     5,6,7,14,3,     1,8,15,16,4,    7,8,9,16,4,     1,7,13,14,4,
6,7,12,4,       1,2,7,14,5,     1,2,9,16,5,     7,12,13,14,5,   7,14,15,16,5,   1,8,9,16,6,
6,7,13,6,       7,8,15,16,6,    6,7,12,13,6,    7,14,21,28,7,   1,2,3,4,7,      7,8,13,14,8,
7,8,15,8,       1,6,7,14,8,     7,8,9,15,8,     1,8,9,15,8,     6,7,8,13,8,     6,7,14,15,8,
6,7,13,8,       1,8,9,10,9,     7,13,14,20,9,   1,2,9,10,9,     6,7,13,20,9,    7,8,15,22,9,
2,6,7,9,        7,14,15,22,9,   1,5,6,7,9,     1,2,3,8,10,     7,14,15,21,10,  5,6,7,8,10,
7,14,21,10,     7,8,14,21,10,   6,7,8,9,10,     7,13,14,21,10,  1,2,3,9,10,    1,7,14,21,11,
2,3,10,11,      1,2,3,7,11,     7,14,21,22,11,  1,8,15,22,11,   4,5,6,7,11,     7,14,20,21,11
8,9,10,11,      1,2,7,8,12,     1,7,8,15,12,    1,7,8,14,12,    1,7,8,9,12,     1,2,8,9,12,
6,7,8,12,       7,8,14,15,12,   6,7,13,14,12,
```

11 Het ECOL programma uit § 10 werd in okt. 1969 in het Elektronisch Rekencentrum te Utrecht door de daar beschikbare vertaler in ALGOL omgezet. Met dit programma waren op de EL-X8 na 6 minuten rekentijd de volgende 24 oplossingen gemaakt:

1:	2	6	22	13	17	31	36	48	18	58	45
2:	2	6	22	13	17	31	36	51	62	18	45
3:	2	6	22	13	51	31	18	44	34	16	59
4:	2	6	22	13	51	31	18	44	56	16	39
5:	2	6	22	24	14	11	38	48	63	18	42
6:	2	6	22	24	14	33	13	62	52	19	45
7:	2	6	22	25	10	17	18	38	43	60	55
8:	2	6	22	25	10	38	18	59	54	17	47
9:	2	6	22	25	17	11	36	48	18	58	45
10:	2	6	22	25	17	11	36	51	62	18	45
11:	2	6	22	25	18	44	16	36	59	10	55
12:	2	6	22	25	18	44	16	36	61	10	53
13:	2	6	22	25	18	44	51	36	12	16	59
14:	2	6	22	25	18	51	44	36	12	16	59
15:	2	6	22	25	18	62	33	46	54	17	11
16:	2	6	22	25	18	62	44	13	52	16	37
17:	2	6	22	25	32	11	59	41	20	15	53
18:	2	6	22	25	32	20	15	11	41	54	59
19:	2	6	22	25	32	20	15	11	51	46	59
20:	2	6	22	25	32	20	15	51	13	46	59
21:	2	6	22	25	32	20	52	44	11	16	59
22:	2	6	22	25	32	51	11	15	46	20	59
23:	2	6	22	25	48	10	20	41	17	37	61
24:	2	6	22	25	48	10	20	41	17	60	32

(De lezer zal gemakkelijk met behulp van de plakjeslijst uit fig. 2 de oplossingen kunnen leggen, mits hij er rekening mee houdt dat in deze oplossingen het merkteken van het kruis gefixeerd is op veld 3).

De snelheid is misschien teleurstellend. Het soort ALGOL dat de ECOL-ALGOL vertaler produceert, en de wijze waarop daarvan weer machinetaal gemaakt wordt zijn voor combinatorische programma's veel minder geschikt dan voor programma's met veel numeriek rekenwerk. Men zou kunnen verwachten dat bij zeer goed overwogen programmering, direct in de machinetaal, op de rekentijd een factor 50, of althans iets van die orde, zou kunnen worden gewonnen.
Velen hebben, onafhankelijk van elkaar, het aantal oplossingen van de 6 x 10 pentomino (met centrum van het kruis, linksboven het centrum van het bordje) vastgesteld op 2339. Zo men wil, kan men dus het totale aantal oplossingen 4 x 2339 = 9356 noemen.
In maart 1963 werd het aan de T.H. Eindhoven gedaan op een machine die men thans klein en langzaam kan noemen (IBM 1620). Het gebeurde met een zeer lang programma in machinetaal, dat zelf grotendeels door de computer zelf (met behulp van een programma-genererend programma) werd gemaakt. Het kostte 18 uur rekentijd.

12 Degenen die vertrouwd zijn met ALGOL zullen misschien liever een ALGOL-programma zien in plaats van het ECOL-programma. Het onderstaande is *niet* het ALGOL-programma dat de ECOL-ALGOL-vertaler van het ECOL-programma maakte. Integendeel, de auteur heeft uit het onderstaande programma (door vertaling met de hand) het ECOL-programma gemaakt.
In dit ALGOL-programma komt de niet-gedeclareerde procedure 'drukoplossinggaf' voor. We laten in het midden op welke manier deze output wordt verzorgd.
De getallenband is dezelfde als bij het ECOL-programma.

```
begin       integer array relpos [1:63, 1:4], stuknr [1:63], bezet [1:77],
            magazijn [1:12], plakstapel [1:12], gatstapel [1:12];
            integer etage, plaknr, eerstegat, i,k,j;
            for k:=1 step 1 until 63 do
                        begin for i :=1 step 1 until 4 do relpos [k,i] := read;
                                    stuknr [k] := read
                        end;
            for j:= 3,9,10,16,17,23,24 do
            begin for k := 1 step 1 until 70 do bezet [k] := 0;
                        for k:= 7 step 7 until 70 do bezet [k] := 1;
                        for k:= 71 step 1 until 77 do bezet [k] := 1;
                        for k:= 0,6,7,8,14 do bezet [j + k] := 1;
                        for k:= 2 step 1 until 12 do magazijn [k] := 1;
                        etage : = 1; eerstegat : = 0;
nieuwgat:               eerstegat := eerstegat + 1;
                        if bezet [eerstegat] = 1 then goto nieuwgat;
                        plaknr := 2;
vulpoging:              if magazijn [stuknr [plaknr] ] = 0 then goto volgendplakje;
                        for i := step 1 until 4 do
                              if bezet [eerstegat + relpos [plaknr, i] ] = 1 then
                                          goto volgendplakje;
                        magazijn [stuknr [plaknr]] := 0; bezet [eerste gat] : =1;
                        for i:= step 1 until 4 do
                              bezet [eerstegat + relpos [plaknr, i] ] := 1;
                        gatstapel [etage] := eerstegat; plakstapel [etage] := plaknr;
                        etage := etage + 1; if etage <12 then goto nieuwgat;
                        drukoplossinggaf; goto poets;
volgendplakje:          plaknr := plaknr + 1; if plaknr ≤63 then goto vulpoging;
poets:                  if etage >1 then

                  begin etage := etage – 1; eerstegat := gatstapel[etage];
                        plaknr := plakstapel [etage];
                        bezet [eerstegat] := 0;
                        for i := 1 step 1 until 4 do
                              bezet [eerstegat + relpos [plaknr,i] ] : = 0;
                              magazijn [stuknr [plaknr] ] := 1; goto volgendplakje

            end
      end
end
```

13 Wij hebben in het voorafgaande geprobeerd het programma begrijpelijk te houden ter wille van de presentatie, en hebben een aantal voor de hand liggende rekentijd-besparende wijzigingen vermeden. Een voorbeeld: als op een gegeven ogenblik het 12^e stukje niet in het magazijn zit, wordt in ons programma 8 keer een plakje geprobeerd, (nl. de plakjes 56 t.e.m. 63). Dat gaat in ons ECOL-programma (en ons ALGOL-programma is wat dit betreft niet beter) via regels 60,61,62,94,95, en dat acht keer! Het is niet moeilijk hier wat tegen te doen.
Een ander geval: Men kan zonder ongelukken de regels 76 en 100 schrappen. Wanneer men nl. een nieuw plakje probeert te leggen (met merkteken op eerstegat) kan dit wèl een vroeger gelegd plakje overlappen, maar dat kan nooit op het merkteken van dat plakje gebeuren.

14 Tenslotte merken we op dat ons zoekproces door middel van opbouw en afbraak een bijzonder geval is van wat men backtracking [4] noemt. Men kan dat beschrijven als het doorlopen van een 'puzzleboom' (hier de boom der testwoorden), maar ook als het sprongsgewijs doorsnuffelen van een woordenlijst, zoals in deze voordracht is gebeurd.

Voetnoten

1 Tekst van een voordracht gehouden in de Cursus 'Computerkunde' op 12 september 1969 te Eindhoven; herhaald op 6 januari 1970 te Utrecht.

2 Voor meer gegevens over deze puzzle en aanverwante puzzles verwijzen we naar: S.W. Golomb, Polyominoes, Charles Scribner's Sons, New York 1965.

3 Een beschrijving van de programmeertaal ECOL is te vinden in: C.A.Ch. Görts, S.G. van der Meulen, A. van der Sluis, J.R. Zweerus, Computerkunde 1, voor a.v.o. en v.w.o., Wolters-Noordhoff, Groningen 1970.

4 Zie 'Computerwiskunde' (red. J.J. Seidel). Aula-reeks nr. 407, 1969. Blz. 75-89.