DOCENTEN
BOEK
COMPUTERKUNDE

Y.20BA-2
(ex 2)

DOCENTEN BOEK COMPUTERKUNDE

IOWO

Instituut Ontwikkeling Wiskunde Onderwijs

RIJKSUNIVERSITEIT UTRECHT
1649 5792

© 1973 Instituut voor Ontwikkeling van het Wiskunde Onderwijs

Niets uit deze uitgave mag worden verveelvoudigd en/of openbaar gemaakt door middel van druk, fotokopie, microfilm of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de houder van het copyright.

inhoud

WOORD VOORAF		blz.	7
HOOFDSTUK 1			
Doelstellingen Computerkunde			
1.1	Computerkunde bij het A.V.O. Prof.Dr. A. van der Sluis	"	8
1.2	Voortgezette computerkunde	"	12
1.3	Internationale aanbeveling	"	15
HOOFDSTUK 2			
Aanwijzingen voor de les			
2.1	De leerstof	"	18
2.2	Tijdschema	"	36
2.3	Opgaven	"	37
2.4	Enkele programma's	"	47
HOOFDSTUK 3			
Het "ECOL-rapport"			
I	Taaldefinitie, vertaling, executie	"	51
II	Globale vorm van ECOL programma's	"	52
III	Contextregels en compileraktie	"	53
IV	Executie	"	56
V	SUB en FUN	"	66
VI	De ECOL syntax	"	72
HOOFDSTUK 4			
Wetenswaardigheden			
4.1	Literatuurbespreking	"	76
4.2	Terminalgebruik bij de computerkunde	"	80
HOOFDSTUK 5			
Achtergrondinformatie			
5.1	Computerhardware	"	82
5.2	Datastructuren	"	89
5.3	Iets over software	"	92
5.4	Recursiviteit	"	97
HOOFDSTUK 6			
Programmaverwerking			
6.1	De ECOL schrapkaart	"	102
6.2	De ALGOL schrapkaart	"	104
6.3	Verzenden van schrapkaarten	"	105
6.4	Kosten programmaverwerking	"	106
6.5	Bestanden	"	108

Woord vooraf

HET EXPERIMENT COMPUTERKUNDE

Sinds 1968 organiseert de Commissie Modernisering Leerplan Wiskunde (CMLW) een experiment in computerkunde. Aanleiding hiertoe was het rapport nummer 3 van deze commissie. Het experiment beoogt de wenselijkheid en mogelijkheid van algemene invoering van deze leerstof te onderzoeken. Deze invoering heeft echter omvangrijke gevolgen ten opzichte van de heroriëntering van docenten en van de organisatie van het gebruik van computers voor het onderwijs. Daarom wordt voorlopig de voorkeur gegeven aan het telken jaren uitbreiden van het experiment en zo toe te groeien naar computerkunde bij het gehele voortgezet onderwijs.

HET DOCENTENBOEK

vanaf 1970 is ten behoeve van de aan het experiment deelnemende docenten een tijdschrift 'Computerkunde IOWO' uitgegeven. Dit docentenboek is ontstaan als bloemlezing uit voorgaande jaargangen van dit tijdschrift gevoegd bij gedeelten van heroriënteringscursussen. De wijze van samenstellen heeft de eenheid van stijl niet bevorderd, maar dat zal minder storen naarmate U het boek meer als naslagwerk beschouwd.

LEERBOEKEN

Uit het voorgaande volgt dat veel van de inhoud van dit docentenboek direkt betrekking heeft op het eerder genoemde experiment. Zo zullen de aanwijzingen voor de les aansluiten op de in het experiment gebruikte leerboekjes, te weten:

'Computerkunde voor het A.V.O.' , deel 1 en 2
C.A.Ch.Görts, S.G.v.d.Meulen, A.v.d.Sluis, J.R.Zweerus
uitgave Wolters-Noordhof

'Werkschrift Computerkunde'
G.A.Vonk uitgave IOWO

In de tekst wordt hiernaar verwezen met de afkortingen 'Compk' en 'Wrks'. Gebruikers van andere methoden zullen met de lesaanwijzingen zeker ook hun voordeel kunnen doen. In het bijzonder zullen de aanwijzingen voor de programmaverwerking ook van toepassing zijn voor de gebruikers van

'Elementaire Computerkunde voor mavo en havo'
W.van den Camp, H.B.Emanuels, J.B.Lubke, P.Th.J.Ploeger
uitgave Meulenhoff Educatief

EDITIE 1971

Bijdragen voor een volgende druk van het docentenboek worden met belangstelling tegemoet gezien, evenals op- of aanmerkingen op deze aflevering. Het adres is dat van het 'Instituut Ontwikkeling Wiskunde-Onderwijs' (IOWO), de uitvoerende instelling van de commissie genoemd in de eerste alinea. Ik dank hierbij al degenen die hun pennevruchten voor dit docentenboek hebben afgestaan.

november 1971

G.A.Vonk
IOWO
Tiberdreef 4
Utrecht

BIJ DE TWEEDE DRUK

Deze editie is globaal gelijk aan de vorige. Gewijzigde omstandigheden hebben enkele aanpassingen noodzakelijk gemaakt, o.a. uitbreiding van §2.1 en het vervallen van de oorspronkelijke paragrafen 4.2, 4.4 en 4.5.

juli 1973

B.H.Knip, G.A.Vonk

1 Doelstellingen computerkunde

1.1 COMPUTERKUNDE BIJ HET ALGEMEEN VOORTGEZET ONDERWIJS

Het experiment computerkunde mag zich inmiddels verheugen op deelname van het beroepsonderwijs. Voor deze vorm van onderwijs is onderstaand artikel niet bedoeld. Zo zal bijvoorbeeld niet altijd een wiskundedocent aanwezig zijn om het onderwijzen van computerkunde tot zijn taak te rekenen. Aan het beroepsonderwijs nu om te bepalen in hoeverre zij van onderstaande beweegredenen moeten afwijken. Het artikel van Prof.Dr. A. van der Sluis is overgenomen uit 'Euclides', jaargang 46 nummer 3.

Inleiding

Dit stuk beoogt uiteen te zetten dat computerkunde een noodzakelijk onderwerp is op elk leerplan voor algemeen voortgezet onderwijs (verder af te korten als AVO, waaronder begrepen VWO), dat een inleiding tot dit vak in het wiskundeprogramma thuishoort, en door de wiskundeleraar gegeven moet worden. Een leerplanschets wordt gegeven. Men kan dit stuk zien als een uitwerking van [1], echter geheel voor verantwoordelijkheid van de auteur.
Het gaat hier dus duidelijk niet om beroepsonderwijs. Dit betekent niet dat op scholen voor beroepsonderwijs computerkunde niet net zo nodig zou zijn. Maar de doeleinden en middelen kunnen daar wel anders zijn.

1 *Doel van het AVO*

Een groot aantal onderwerpen komt op zichzelf genomen in aanmerking om in het schoolprogramma opgenomen te worden. Het is daarom duidelijk dat men zeer scherpe criteria zal moeten aanleggen alvorens een nieuw onderwerp op te nemen, temeer waar dit over het algemeen zal impliceren dat iets anders moet wijken. Overigens dienen dezelfde criteria te worden aangelegd om te bezien of dat wat is, kan blijven.
Deze criteria hangen uiteraard samen met de doelstellingen van het onderwijs als geheel. Aangezien het, voor zover ik heb kunnen nagaan, al weer 10 jaar geleden is dat in dit tijdschrift aandacht gegeven is aan de algemene doelstellingen van het onderwijs (n.l. in het artikel van W. Peremans, Euclides **36** (1960), p. 129–164), lijkt het me verantwoord hierop allereerst in te gaan. [1]) Temeer waar er inmiddels een mammoetwet in werking is getreden waarin niets over deze materie te vinden is, en er in het voorjaar 1968 voorstellen leerplan rijksscholen zijn verschenen waar het wiskundeleerplan een van de weinige is die niet met een doelstelling ingeleid worden. Het is kennelijk een heet hangijzer, begrijpelijk overigens gezien de snelheid waarmee onze maatschappij en maatschappijbeschouwing evolueren, en daarmee de inzichten betreffende het doel van het onderwijs. Desondanks: hoe kan men leerplannen opstellen zonder een welomschreven doel?
Zeer schematisch gezegd zie ik als belangrijkste taken van het AVO de leerlingen voor te bereiden op
(a) *het leven in de wereld van morgen*
(b) *het werken in de wereld van morgen*
(c) *verdere studie*
Aan elk van deze taken is inherent persoonlijkheidsvorming van de leerling, waaronder begrepen zaken als het bijbrengen van verantwoordelijkheidsgevoel, zin voor samenwerking e.d. Aan dit aspect ga ik in deze beschouwing voorbij, o.a. omdat het me voorkomt dat dit minder met het leerplan dan met de wijze van onderwijs geven samenhangt.
Over elk van deze taken een paar woorden:

Taak (*a*). De leerling moet leren hoe een samenleving functioneert, hoe deze zich ontwikkelt, welke krachten daarin een rol spelen, en hoe deze krachten beheerst kunnen worden.

Taak (*b*). Voor elke taak zijn zekere specifieke kennis en bekwaamheid vereist. Het is niet de taak van het AVO deze bij te brengen, maar wel moet het er de basis voor leggen. Maar ook en vooral moet het algemene bekwaamheden en niet-specifieke kennis bijbrengen. Onder algemene bekwaamheden versta ik bekwaamheden als: tot de kern van een zaak door te dringen, te leren, je kennis te gebruiken, een werk af te ronden, te organiseren. Nu specifieke kennis en bekwaamheid aan een zo snelle veroudering onderhevig zijn wordt de betekenis van deze algemene bekwaamheden steeds groter. De hoedanigheid van een vak, die de ontwikkeling van deze algemene bekwaamheden bevordert, behoort tot de vormende waarde ervan, een vaag begrip waarmee men gewoonlijk de mate aanduidt waarin dit vak in staat is een wenselijke habitus aan te kweken, iemand te leren om andere dingen die met het betrokken vak in feite niets uitstaande hebben, beter te doen. Een vak dat traditioneel hoog om zijn vormende waarde wordt aangeslagen is de wiskunde.

Taak (*c*). In principe verschilt deze taak niet van de vorige, maar in zijn implicaties wel degelijk, doordat de verdere studie wel heel specifieke eisen kan stellen (men denke aan latijn en wiskunde als noodzakelijke voorbereiding voor tal van studierichtingen), waaraan zonder de nodige aandacht op school niet voldaan kan worden, en ook diverse algemene bekwaamheden krijgen hier speciale nadruk.

Het spreekt vanzelf dat deze taken evenzovele criteria vormen die men dient te hanteren bij het overwegen of een onderwerp een plaats op het leerplan verdient of niet. In verband met het grote belang ervan licht ik echter het onderdeel vormende waarde uit (b), en maak hiervan een apart criterium, criterium (d).
Het belang van onderwijs in een bepaald vak is een soort gewogen som van de maten waarin het aan al deze criteria voldoet. Hoe moeilijk deze dingen ook te wegen zijn, het is duidelijk dat opstellers van schoolprogramma's de plicht hebben te streven naar optimalisatie van de output. Dit geldt voor elk te onderwijzen vak apart, maar ook voor het onderwijspakket als geheel,vooral wanneer, zoals ten onzent, de leerling althans in de eerste 3 of 4 jaren bij het AVO slechts de keus heeft uit een paar standaardpakketten.
Heeft men zich er bij de opstelling van het leerplan voor enig vak, bijv. voor de wiskunde, inderdaad bewust rekenschap van gegeven in welke mate elk onderdeel ervan de bovengenoemde doeleinden dient, en in welke mate dezelfde doeleinden met minder moeite bereikt zouden kunnen worden door een andere (zij het wellicht minder bij de traditie of de universitaire mode aansluitende) keuze van de stof?

2 *Het belang van computerkunde*

In het licht van de voorafgaande beschouwingen willen we nu het vak computerkunde bezien. We zullen ons hierbij ter wille van lengte en overzichtelijkheid van dit artikel zeer moeten beperken; elk criterium is op zich goed voor een heel artikel.
Criterium (*a*). Het patroon van onze samenleving ontwikkelt zich in snel tempo en de computer is hierin een belangrijk instrument. Reeds nu heeft de computer vele belangrijke veranderingen teweeg gebracht, terwijl hij nog pas 10 jaar op enige schaal in gebruik is. Slechts zeer weinigen realiseren zich (of moet ik zeggen: hebben genoeg fantasie) in welke mate de computer in de toekomst ons leven zal gaan beinvloeden. We kunnen ons aan deze kracht niet onttrekken, en moeten er ook niet bang voor zijn, maar hem accepteren en trachten ermee vertrouwd te raken, juist als met andere belangrijke krachten in de samenleving.
De computer brengt een sociale en culturele revolutie teweeg, waarvan het belang vergeleken kan worden met dat van de industriële revolutie in de vorige eeuw. Werd toen routinematige *lichamelijke* arbeid door machines overgenomen, nu is dit het geval met routinematige *geestelijke* arbeid. Het feit dat veel activiteiten die altijd van intellectuele aard geacht werden nu routinewerkzaamheden blijken te zijn, die men kan programmeren voor een computer, is ook conceptueel een gebeurtenis van de eerste orde, die een nieuw licht werpt op menselijke activiteiten.
Het huidige gebrek aan inzicht bij de doorsnee burger omtrent de wijze waarop deze nieuwe kracht in onze samenleving functioneert wekt verwarring en onzekerheid. Maar ook bevordert dit dat deze kracht ons gaat overheersen in plaats van andersom. In dit verband mag nog eens aan Norbert Wiener's bekende

[1] Na het gereed komen van dit artikel verscheen juist het artikel 'De doelstellingen van het wiskundeonderwijs' van H. van der Hak (Eucl. **45** (1970), p. 289–299). Het is niet de bedoeling van het huidige artikel met de heer van der Hak in discussie te treden (auteur)

Wij wijzen ook op 'Naar een nieuw onderwijsprogramma voor de wiskunde' van J. van Dormolen (Eucl. **46** (1970), p. 1–7) (redactie).

reactie herinnerd worden op de vraag of het gevaar dat robots ons zouden gaan overheersen niet tamelijk denkbeeldig is omdat men dan toch eenvoudig de stekker uit het stopcontact kan trekken. Wiener's antwoord was dat dit middel wel eens erger dan de kwaal zou kunnen zijn; immers, naar mate men meer aan de robots overlaat, wordt de samenleving meer ontwricht wanneer men ze buiten gebruik stelt. Computers in de hand te houden zal een belangrijke taak zijn van deze generatie, en moet niet worden overgelaten aan een kleine clan van computermagiërs.

Criterium (*b*). Wat onder (a) gezegd werd over de revolutie die computers teweeg brengen raakt natuurlijk ook en heel speciaal de werksituatie.
Vele vroeger door mensen vervulde functies worden nu verricht door computers. Een computer kan sneller en vollediger alle mogelijkheden overzien om een machine of wagenpark te benutten dan mensen. Een computer kan de toestand van een bedrijf op de huidige minuut weergeven. Aan te houden voorraden kunnen veel nauwkeuriger beheerst worden. Bibliografische gegevens kunnen veel sneller teruggevonden worden. Maar ook kan de computer allerlei dingen doen die vroeger helemaal niet gebeurden, zoals het door simulatie schatten van het effect van maatregelen die men overweegt te nemen.
De taken die door mensen moeten worden vervuld verschuiven hierdoor uiteraard aanzienlijk. Wegens de universele bruikbaarheid van de computer kan men rustig zeggen dat een groot percentage van de huidige scholieren in hun latere werkkring op een of andere manier met computers te maken krijgt. Zij zullen hun werk verrichten door middel van computers, (ook al zullen ze hem misschien niet zelf programmeren of bedienen) en zij zullen werk uitvoeren dat geheel of gedeeltelijk door computers is voorbereid. Het is daarbij niet van belang dat zij precies weten hoe men de meest uiteenlopende dingen met een computer doet. Het gaat erom dat zij er besef van hebben wat een computer zoal kan doen, zodat zij in voorkomende gevallen naar een een computerexpert kunnen gaan.
Dit besef, niet de vakbekwaamheid is wat alle leerlingen op school moeten opdoen.

Criterium (*c*). Aangezien in zoveel beroepen de computer een belangrijke rol gaat spelen is het duidelijk dat vele leerlingen de computer bij hun verdere opleiding zullen tegenkomen. Dit is zeker ook waar voor studie aan de universiteit.
Reeds in de nabije toekomst zullen er slechts weinig studierichtingen zijn waarin men niet in meerdere of mindere mate met de computer in aanraking komt. Het zijn bepaald niet alleen de studenten in de exacte wetenschappen die tijdens hun studie met de computer moeten werken, maar ook de studenten in vakken als rechten, humaniora en literaire wetenschappen. In een officieel rapport uit de USA (zie [2]) schat men dat van deze laatste categorieën omstreeks de helft enig computerwerk zullen doen.

Criterium (*d*). De vormende waarde van het leren werken met een computer kan nauwelijks overschat worden. Het kweekt n.l. een sterk operationele, algoritmische en organisatorische instelling aan. Met deze termen bedoel ik het volgende.

Operationeel. Een operationele instelling is het streven kennis en informatie die tot je komt (en die waarover je reeds beschikt) operationeel te maken, d.w.z. geschikt om mee te werken. Dit betekent een evalueren ervan, en je bewust maken wat het impliceert, wat je er aan hebt. De leerlingen blijken vaak wel allerlei dingen te weten, maar het schort steeds weer aan deze operationaliteit. Een gebrek hieraan trachten ze dan vaak voor zichzelf en hun leraar te verbergen achter breedsprakigheid, en vaak schoppen ze het hiermee een heel eind. Bij een computer komen ze echter met breedsprakigheid niet ver. Dan moeten ze zich precies realiseren wat ze weten, wat er gegeven is, en hoe ze hiermee moeten handelen om een gesteld doel te bereiken.

Algoritmisch. Het woord algoritme betekent zoiets als een rekenwijze. Algemener verstaan we onder algoritme elk eenduidig voorschrift om een proces (bijv. een rekenproces of een breiproces of een telefoneerproces) uit te voeren met eindig veel welbegrepen en uitvoerbare handelingen. Algoritmiek is dan de discipline om algoritmen te herkennen, te construeren en te analyseren. Deze algoritmen moeten niet alleen tot stand brengen wat van hen verlangd wordt, maar moeten gewoonlijk ook nog aan andere condities voldoen zoals eenvoud en efficiency. Een algoritmische instelling is het streven de routineaspecten in allerlei processen te ontwaren, algoritmen voor gegeven processen te ontwikkelen.
Het belang hiervan is duidelijk. Processen die creativiteit vereisen kunnen slechts worden uitgevoerd door enkelingen, routineprocessen kunnen worden uitgevoerd door velen. En deze processen kunnen dan als bouwstenen gebruikt worden voor meer gecompliceerde processen. Dit mechanisme, creativiteit om te zetten in routine, ligt ook ten grondslag aan veel wetenschappelijke vooruitgang. Wat gisteren nog moeilijk was en veel hoofdbrekens kostte is vandaag routine geworden zodat men zich kan concentreren op verder reikende problemen. Er zijn tal van voorbeelden dat de ontwikkeling van de wetenschap langdurig stagneerde doordat allerlei kennis die al wel vergaard was nog niet tot een goed bruikbaar apparaat was omgesmeed (men denke zich maar eens in dat wij nog zo moesten rekenen zoals de Romeinen het deden of dat wij moesten integreren zoals Archimedes). Een mens is nu eenmaal niet in staat om veel problemen tegelijkertijd te behandelen; hij moet ze inkapselen, hij moet verdelen om te kunnen heersen. Dit is ook van belang in de school. Er is tegenwoordig een verschuiving van routine naar wat begrip wordt genoemd. Routine is haast een smerig woord geworden. En inderdaad is er in het verleden teveel onbruikbare routine en te weinig bruikbaar begrip aan de leerling gepresenteerd. Routine moet functioneel zijn, het moet de apparatuur zijn waar de leerlingen mee werken en dan is het onmisbaar.
Routine te vinden, te isoleren en als algoritme te formuleren is een creatieve activiteit. Een algoritmische instelling is hiervoor een vereiste.

Organisatorisch. Na het voorafgaande kunnen we hierover kort zijn. Werken met een computer betekent organiseren. Bij organiseren is het belangrijk niveau's van complexheid te onderscheiden, alle mogelijke gevallen te voorzien en schijnbaar verschillende zaken onder één noemer te brengen. Dit is precies wat werken met de computer aankweekt.

3 *Wat voor computerkundeonderwijs*

Veel onderwijs vaart onder valse vlag. Iedereen beaamt dat wiskunde een belangrijk vak is en is bereid dit zo nodig met een aantal voorbeelden te staven. Het is helemaal niet moeilijk om voor de wiskunde een soortgelijke beschouwing te houden als hierboven onder 2 gedaan is voor computerkunde. Desondanks zat (en zit?) in het wiskundeprogramma veel dat niet aan de in 1 genoemde criteria voldoet.
Evenzo kan men ook zaken onder de vlag van belangrijkheid van de computerkunde laten meevaren waarop het onder 2 gestelde niet of veel minder van toepassing is. Velen zijn van mening dat bij computerkunde het binaire stelsel, flipflops en magneetringetjes belangrijke onderwerpen zijn, en sommigen menen dit zelfs ook ten aanzien van het bedrijfssysteem van de computer (dat de samenwerking tussen de diverse componenten regelt), datarepresentatie en -opslag, compilers etc.
Ik wil hier duidelijk stellen dat onderwijs in het binaire stelsel, de computerhardware en de computersoftware niet vallen onder de belangrijkheid die onder 2 voor computerkunde werd geclaimd. Deze onderwerpen zijn van tweede importantie, bezien in het kader van het onderwijs als geheel. Wat wel belangrijk is is algoritmiek (zie 2). Het gaat er om dat de leerling in staat is het principe van grote computertoepassingen te doorgronden en (kleine) eigen probleempjes m.b.v. een computer op te lossen.
In tegenstelling tot wat nogal eens gezegd wordt kan dit onderwerp uitstekend behandeld worden zonder dat de leerling inzicht heeft in computerhard- en software. In feite hebben ook de meeste gebruikers van computers totaal geen idee van deze zaken. Wie dit verbaast denke slechts aan auto's en TV toestellen, ook ingewikkelde apparaten die door velen gebruikt worden zonder te weten hoe ze werken.
Evenmin is het juist dat de leerling pas interesse voor computertoepassingen kan opbrengen wanneer hij weet hoe het apparaat werkt. Ervaring in de klas toont duidelijk aan dat het leeuwendeel der leerlingen er zeer in geinteresseerd is te weten hoe je met een computer werkt en er ook inderdaad mee te werken ook al weten ze niet hoe de computer werkt.
Daarom, wanneer men dan al iets aan computer hard- of software wil doen dan dient dit uitsluitend ter bevrediging van de nieuwsgierigheid van de leerling, en staat als zodanig op hetzelfde vlak als het uitleggen hoe een televisietoestel werkt.
Door de voornaamste aandacht aan algoritmiek te geven wordt de omvang van het vak computerkunde tot veel kleinere proporties teruggebracht dan men wel eens geneigd is te denken. En dat is maar goed ook, want op school is tijd een even kostbaar goed als overal elders.
Het is essentieel dat de leerling ook inderdaad met een computer werkt althans in deze zin dat hij programma's schrijft die op een echte computer verwerkt worden. Anders blijft het maar bij droogzwemmen en dit is weinig inspirerend en erg onbevredigend voor de leerling. Pas wanneer hij resultaten van de computer terug krijgt vindt het wonder plaats dat deze machine, die eens met zoveel respect bekeken werd, op zijn juiste plaats gezet wordt: een slaaf van de mens en zelfs van een schooljongen. Een tweede reden waarom het essentieel is dat de leerling programma's schrijft en de resultaten terug krijgt van een computer is om hem gevoel te geven voor de relativiteit van resultaten die met een computer verkregen zijn. Op het ogenblik is het nog zo dat voor velen een bewering, dat de computer iets heeft geconstateerd, geen tegenspraak duldt. In werkelijkheid zijn de resultaten niet van beter gehalte dan degeen die het programma ontwierp. Het is belangrijk dat de leerling zelf ziet hoe gemakkelijk men gevallen over het hoofd ziet en welke grote consequenties kleine programmeerfouten kunnen hebben.

Inmiddels moet men oppassen dat het programmeren niet teveel nadruk krijgt of nog erger dat het hele vak tot een programmeercursus ontaardt. Het is niet nodig dat de leerling een erg goed programmeur wordt. Zijn programma's hoeven geen voorbeelden van efficiency te zijn (alhoewel het ook geen voorbeelden van inefficiency hoeven te zijn). Het komt me voor dat een juiste instelling voor de leraar is dat hij het programmeren eigenlijk als een noodzakelijk kwaad beschouwt.
De te gebruiken programmeertaal dient dan ook met grote zorg gekozen te worden. Deze hoeft enerzijds niet al de verfijning van ALGOL of FORTRAN te hebben, maar moet anderzijds ook zeker de leerling niet opschepen met al de administratieve rompslomp van een machinetaal. Er moet prettig met variabelen, rijen en aritmetische uitdrukkingen (bijv. $(a+b)/2$) gewerkt kunnen worden. Om de eerste stappen van de leerling op het computerpad zo eenvoudig mogelijk te maken hebben wij het verantwoord geacht een aparte taal te ontwikkelen, ECOL geheten (Educational Computer Language), die min of meer in de doorsnee van ALGOL en FORTRAN ligt, maar een aantal onaangenaamheden van deze talen mist ten koste van wat efficiency (bijv. is er geen declaratie van enkelvoudige variabelen en er zijn geen types) en die ook wat minder kansen op vergissingen geeft (de slordigheid van leerlingen bij het schrijven van programma's is ongelooflijk). Het bezwaar dat deze taal ECOL niet algemeen gebruikelijk is en dat de leerling liever een van de bestaande talen zou moeten leren omdat dat hetgene is wat hij later zal nodig hebben, vonden wij niet erg groot: als je eenmaal een programmeertaal kent is het leren van een volgende niet moeilijk meer. Bovendien komen en gaan programmeertalen, en is er alle kans dat de leerling zich later moet gaan bedienen van een programmeertaal die er nu nog niet eens is.
De toepassingen verdienen veel aandacht. Niet alleen de kleine toepassingen die gemakkelijk programmeerbaar zijn, maar ook en vooral de grote toepassingen rondom ons heen waaruit het belang en de interessantheid van computers duidelijk wordt. Dit zijn de toepassingen waarop criteria (a) en (b) van toepassing zijn. Dit ook zijn de problemen waar algoritmiek in zijn ware setting aan het licht treedt, en dit is niet het geval met kleine programmeeropgaven.
Bij deze grotere toepassingen komt men in aanraking met de kwestie van het maken van een *model* van de werkelijkheid. Men kan geen berekeningen maken aan de werkelijkheid zelf, maar de werkelijkheid moet *gerepresenteerd* worden door een computermodel en aan dit model worden de berekeningen gemaakt. Dit heeft twee belangrijke implicaties. De eerste betreft weer de relativiteit van computerresultaten: deze zijn ook niet beter dan het model waaraan de berekening plaats vond. Het model stelt immers hoogstens een *deel* van de werkelijkheid voor en elke invloed die dit deel van de werkelijkheid ondergaat van gebeurtenissen er buiten kunnen niet in de berekende resultaten weerspiegeld worden. De tweede implicatie ligt in het vlak van criterium (d). Een bekwaamheid om relevante modellen van de werkelijkheid te bouwen is belangrijk bij vele onderwerpen op school en daarna. Modelbouw is niet eenvoudig. Het betekent onderscheiden wat belangrijk is (en daarom in het model moet worden opgeomen) en wat niet (en kan worden weggelaten). De aantrekkelijkheid van deze modelbouwerij in de computerkunde is de nauwe relatie ervan met het alledaagse leven en het feit dat men er geen moeilijke theorieën bij nodig heeft (zulks in tegen- stelling tot de situatie in bijv. natuurkunde en scheikunde).
Samenvattend: computerkunde bij het AVO moet zich bezighouden met het organiseren van processen met een computer; er zijn dan zeer wenselijke neveneffecten.

4 *Schets leerplan voor computerkunde*

De voorafgaande beschouwingen hebben ons tot een leerplan geleid dat ik hier slechts beknopt wil schetsen. Nadere uitwerking ervan vindt men in [3].

(I) *Elementaire algoritmische begrippen en hun realisatie op een computer.*
(a) geheugen en waardetoekenning
(b) aritmetische uitdrukkingen
(c) in- en output opdrachten
(d) vertakking en samenvloeiing
(e) cyclus
(f) rij.
Deze onderwerpen dienen zowel in termen van blokschema's als van een eenvoudige programmeertaal gestalte te krijgen en aan tal van eenvoudige voorbeelden te worden verduidelijkt; door de leerlingen zelf geschreven programma's dienen op een computer verwerkt te worden. Dit deel I zou zoiets als 20 lessen moeten beslaan.

(II) *Algoritmen van hoger plan.*
Dit omvat het opstellen van blokschema's voor problemen waarvoor complete programma's te lang zouden worden. De blokken van deze blokschema's stellen elk op zich weer een algoritmisch proces voor waarvan de leerling zich op dit moment kan voorstellen dat er een programma voor te schrijven is of waarvoor misschien in het voorafgaande al programma's geschreven zijn.
In deze samenhang zou de subroutine ter sprake kunnen komen maar we hebben daarin niet veel nut gezien. De subroutine vergt vrij wat uitleg, en het is moeilijk om voorbeelden te bedenken waarin de subroutine op voor de leerling relevante wijze gebruikt wordt.
Ook is nu het moment aangebroken om wat meer aandacht te geven aan algoritmen die helemaal niet computer-georiënteerd zijn maar aan het dagelijks leven ontleend.

(III) *Principes van belangrijke computertoepassingen.*
(a) personeelsadministratie of burgerlijke stand
(b) bank- of girosysteem
(c) reserveringssysteem
(d) simulatie
(e) kunstmatige intelligentie
(f) operaties op teksten (string handling).
Alleen de principes zijn van belang. Een compleet giroprogramma zal wel duizenden opdrachten bevatten. Maar het principe kan uitgelegd worden met een programma van twintig opdrachten. Een meer verfijnd girosysteem kan dan besproken worden met stroomdiagrammen zoals onder (II) is uiteengezet, programmeeropgaafjes kan men aan de blokken van deze stroomdiagrammen ontlenen en op deze wijze krijgen de leerlingen heel wat inzicht zonder overdreven programmeerinspanning. Overeenkomstige opmerkingen kunnen gemaakt worden t.a.v. de andere toepassingen. Dit is ook een indicatie dat men de onderdelen (II) en (III) tezamen zou kunnen doen.

Dit is klaarblijkelijk een erg bescheiden programma. Toch verwachten we dat dit programma, dat minder dan 2 jaaruur vergt, duidelijk inzicht en bekwaamheid aan een groot aantal leerlingen kan geven. Dit laatst kan zeker niet gezegd worden van vele veel kostbaarder onderwerpen. Dit plaatst computerkunde in een unieke en attractieve positie.

5 *Wie onderwijst computerkunde en aan welke leerlingen*

Het is een tamelijk controversiële vraag wie de juiste man (of vrouw) is om computerkunde te onderwijzen en ook welke status dit onderwerp in het schoolprogramma moet hebben.
Sommigen menen dat het een apart vak met een aparte docent moet zijn. Naar mijn mening moet dit echter beslist niet. Er moeten niet teveel verschillende onderwerpen op school zijn. Al te lang hebben we gezucht onder een groot aantal vakken en vakjes die duidelijk samen hangen zoals geschiedenis, staatsinrichting en economie; natuurkunde en mechanica; algebra en meetkunde; maar waarvan de kennelijke samenhang niet uit de verf kwam doordat ze als verschillende vakken en door verschillende leraren gegeven werden. Gelukkig is er tegenwoordig eerder een streven naar integratie dan naar differentiatie en dat moet niet door computerkunde doorbroken worden. Juist niet door computerkunde, want als er één vak is dat relaties met vele andere vakken heeft dan is het wel computerkunde.
Dit laatste argument laat velen in de andere richting doorslaan. Men moet het laten geven van computerkundeonderwijs niet beperken tot leraren van een bepaald vak, en in het bijzonder niet tot de wiskundeleraren; wanneer de leraar Frans iets van computers afweet moet hij er maar over vertellen, en idem de leraar scheikunde etc., zo zeggen zij.
Tot op zekere hoogte ben ik het hier wel mee eens. Het is inderdaad zeer belangrijk dat in vele vakken het mogelijke gebruik van computers duidelijk gemaakt wordt, omdat de leerling in zijn latere leven de computer in zo velerlei samenhang zal tegenkomen. Het is ook zeer belangrijk omdat een vak meer operationeel (zie 2) voor de leerling wordt en meer reliëf krijgt wanneer hem de computerimplicaties van het vak getoond worden. Vanzelfsprekend hangt dit in sterke mate af van de toevallige leraar die een vak onderwijst.
De vraag is echter: wie zal de leerling de eerste stappen naar de computer laten zetten? In welk vak zal deze inleiding een plaats krijgen? Want het is beslist ongewenst om deze inleiding te laten afhangen van de toevallige leraren die de verschillende vakken hebben. Hoe eenvoudig deze inleiding ook moge zijn (zie 4), het is een onderwerp met een eigen structuur en methodologie, en iemand die wat van programmeren af weet en weet hoe een computer te gebruiken voor zijn eigen doeleinden is niet ipso facto gekwalificeerd om deze inleiding te onderwijzen.
Het is duidelijk dat van alle vakken op school de wiskunde het vak is dat qua structuur en methodologie het nauwst verwant is met een inleiding tot computerkunde zoals in 4 geschetst. Ook is het waar dat de wiskundeleraar het meest ervaren is om dit soort manipulaties en structuren te onderwijzen en daarmee ook om gebrek aan begrip bij de leerlingen op te sporen en te verhelpen.
Het onderwijs in de principes van de toepassingen zie ik graag in een hand, om de daarbij optredende samenhang aan het licht te brengen, en weer is het de wiskundeleraar die ik dit het liefste zie doen. Hij houdt op waar het interessant

begint te worden voor de professional in wiens vakgebied de toepassing ligt. Dit opent de mogelijkheid voor een gelukkige samenwerking en taakverdeling.
Het zou echter ook heel goed zijn voor het wiskundeprogramma zelf als de computerkunde daarin een plaats kreeg. Het wiskundeprogramma immers neigt steeds meer in een abstracte richting; deze realiteitsinjectie kan een sterke impuls in een meer toegepaste richting geven.
Om deze redenen ben ik van mening dat het gerechtvaardigd is te trachten de computerkunde in het wiskundeprogramma onder te brengen.
Hierbij rijst natuurlijk vanzelf de vraag of dan de wiskunde extra uren moet krijgen, en waar die vandaan moeten komen. Ik vraag me evenwel af of dit inderdaad nodig is.
Enerzijds dient men hierbij te overwegen dat de wiskunde altijd voor een belangrijk deel gemotiveerd is door zijn vormende waarde. Computerkunde heeft een vormende waarde die deels in hetzelfde vlak ligt als die van de wiskunde, maar de vormende waarde van computerkunde komt naar mijn overtuiging beter tot zijn recht door de veel grotere mogelijkheden voor creatief bezig zijn die dit vak de leerlingen biedt.
Voorts werd de wiskunde vooral gemotiveerd door haar toepasbaarheid. In diverse toepassingsgebieden heeft de computer evenwel de wiskunde verdrongen. Een en ander maakt dat een claim van de computerkunde op wiskunde uren zeker overwogen dient te worden, mede in het licht van de opmerkingen over optimalisatie aan het eind van 1.
Anderzijds zou het mij niet verwonderen wanneer computerkundeonderwijs een zodanig gunstige invloed op de instelling van de leerlingen heeft (zie hierboven onder criterium (d)) dat, ook al neemt men de tijd voor computerkundeonderwijs uit de wiskunde-uren, toch het wiskundeprogramma niet hoeft te worden ingekrompen. Er zijn reeds enkele positieve indicaties in deze richting Het zou zeer wenselijk zijn wanneer de leraren die nu reeds met computerkunde experimenteren, over dit punt hun licht eens zouden willen laten schijnen.
Resumerend: het is zeer gewenst dat de wiskundeleraar de inleiding geeft, en dat alle andere leraren maximale aandacht aan het verschijnsel computer schenken in hun lessen.

Tenslotte nog de vraag aan wie computerkundeonderwijs gegeven moet worden. In het voorafgaande ligt besloten dat alle leerlingen dit onderwijs moeten krijgen en wel in een niet al te laat leerjaar, dit laatste vooral in verband met de vormende waarde van het vak. Wij denken aan hoogstens de derde klas van het AVO.

6 *Ervaringen*

De ervaringen in de klas zijn nog gering. Wat we tot dusverre hebben gezien stemt ons echter optimistisch. De leerlingen zijn in grote meerderheid enthousiast, en komen tot niet vermoede activiteiten. Eigen problemen worden bedacht, en er ontstaat teamwork. Wij zouden het zeer op prijs stellen als ervaringen van tijd tot tijd een plaats in dit tijdschrift zouden vinden.

LITERATUUR

[1] Rapport over de wenselijkheid en mogelijkheid van het invoeren van computerwiskunde in het onderwijs voor MAVO, HAVO en VWO. Commissie Modernisering Leerplan Wiskunde, Utrecht 1968.
[2] Computers in higher education. Report of the President's Science Advisory Committee, Washington 1967 (het zgn. Pierce report).
[3] C. A. C. Görts, S. G. van der Meulen, A. van der Sluis, J. R. Zweerus: Computerkunde voor AVO en VWO. Wolters-Noordhoff, Groningen 1970.
[4] A. van der Sluis: Onderwijs in computerkunde. In: J. H. Wansink, Didactische Oriëntatie voor wiskundeleraren, deel III, p. 276–315, Wolters-Noordhoff, Groningen 1970.

1.2 VOORTGEZETTE COMPUTERKUNDE

1.2.1 Basiscursus

In de voorgaande paragraaf is het hoe en waarom van het leervak computerkunde uiteengezet. Dit betreft een algemeen vormend leervak dat, volgens 73% van de deelnemende docenten in 1971, twee wekelijkse lesuren gedurende één leerjaar zou moeten beslaan. Kort samengevat zijn de belangrijkste oogmerken van deze cursus:

- voorbereiding op de samenleving, waarin de computer een steeds toenemende rol zal spelen. Het is aan ons een generatie op te voeden, die zal zorgen dat dit een dienende rol zal blijven.
- voorbereiding op toekomstig werk, met het besef welk deel van de werkzaamheden aan computers kan worden overgelaten.
- voorbereiding op toekomstige studie, waarin vrijwel zeker computertoepassingen op wetenschappelijk terrein voorkomen.
- aankweken van een operationele instelling, d.w.z. een instelling waarmee kennis ook geschikt gemaakt wordt om mee te werken.
- aankweken van algoritmische vaardigheid, d.w.z. het vermogen om schijnbaar chaotische systemen te ordenen rond routinematige kernen.
- aankweken van organisatorische kwaliteiten, d.w.z. het onderscheiden van niveaus van complexheid, het voorzien van alle mogelijke gevallen en het onder één noemer brengen van schijnbaar verschillende zaken.
- Inzicht verstrekken in het bouwen van een model, geen berekeningen makend aan de werkelijkheid zelf, maar aan een computermodel dat deze werkelijkheid representeert.

In de doelstellingen die tot nu toe zijn uiteengezet wordt vermeld dat het leervak bedoeld is voor *alle* leerlingen van het algemeen voortgezet onderwijs. Ik zou daar graag het lager beroepsonderwijs aan willen toevoegen. Of dit leervak apart of geïntegreerd in wiskunde, praktische beroepsvorming of welk ander vak ook, moet worden gegeven, wordt verschillend beoordeeld.
Opgenomen in bijvoorbeeld de wiskunde heeft het bezwaar dat computerkunde niet de haar toegemeten tijd krijgt. Te veel bij een leraar met computerhobby, te weinig bij een overladen programma.
Een apart leervak computerkunde druist in tegen de moderne opvattingen van integratie van leervakken. Vraag is in dat geval ook welke leraar bevoegd moet worden geacht voor het doceren van computerkunde.

1.2.2 Voortzetting van de basiscursus ?

Bij veel docenten komt de vraag op of na beëindiging van de basiscursus het leervak niet kan worden voortgezet in de hogere leerjaren, mogelijk zelfs een eindexamenvak kan worden. Dit wordt mede gevoed door het feit dat in veel gevallen binnen het experiment de computerkunde gedoceerd werd als keuzevak in de vierde klas HAVO. Andere keuzevakken kunnen vaak voortgezet worden in de vijfde klas HAVO.
Welke inhoud zou moeten worden gegeven aan deze voortzetting? Bij de beantwoording van deze vraag zijn m.i. zes aspecten te onderscheiden. Van de eerste vijf kunnen kombinaties voorkomen, maar hieronder worden ze zoveel mogelijk disjunct aangegeven. Getracht wordt aan te tonen dat alleen het zesde aspect aanvaardbaar is voor algemeen voortgezet en lager beroepsonderwijs.

1) ***Apparatuur*** (hardware)
De constructie en de electronische werking van de computer, bijvoorbeeld de werking van een kernengeheugen, binaire code, poorten, logische schakelingen.
Deze leerstof heeft voor vele docenten met een beetje technisch gevoel,en ik reken mezelf hier bij, een grote charme. Wiskundedocenten die weleens hun natuurkunde-collega's hebben benijd om de mogelijkheid hun leerlingen in een praktikum te laten werken, zien nu een mogelijkheid om met 'computerbouwdozen' hiermee te wedijveren.

Vragen we ons af welk doel we dienen met deze leerstof, dan moeten we meestal het antwoord schuldig blijven. De kans dat we een toekomstig computertechnicus onder onze leerlingen tellen is minimaal. Kennis die de leerlingen hebben opgedaan is niet toepasbaar en is in alle gevallen een zo sterke simplificatie, dat de werkelijkheid nauwelijks meer gedekt wordt. Als U vindt dat Uw leerlingen toch wat Boole-algebra hebben opgestoken, dan hebt U het paard wel achter de wagen gespannen.
Willen Uw leerlingen (althans technisch aangelegde jongens) zo graag weten hoe het apparaat werkt?
Maar ook de meest kritische leraar zal zijn lessen niet alleen afstemmen op wat zijn leerlingen het liefst horen. En, nogmaals, de constructie van een computer kan niet worden behandeld zonder de leerlingen een rad voor de ogen te draaien.

2) *Harde programmatuur*
Dit is het programmapakket dat altijd in de computer aanwezig is om het apparaat prettig bruikbaar te maken. In werkelijkheid zal de opdracht a := LEES een programma aktiveren, dat een getal cijfer voor cijfer leest, een toestandsteken en eventuele decimaalpunt op de juiste wijze interpreteert en het getal opbergt in de bedoelde geheugenplaats. Geen enkele gebruiker van een computer maakt zich het bestaan van harde programmatuur voortdurend bewust. De computerleverancier dient zich hierover zorgen te maken. Het is niet de bedoeling van ons onderwijs om personeel van computerfabrikanten op te leiden. Deze leerstof zal vrijwel niet kunnen bogen op belangstelling van de leerlingen.

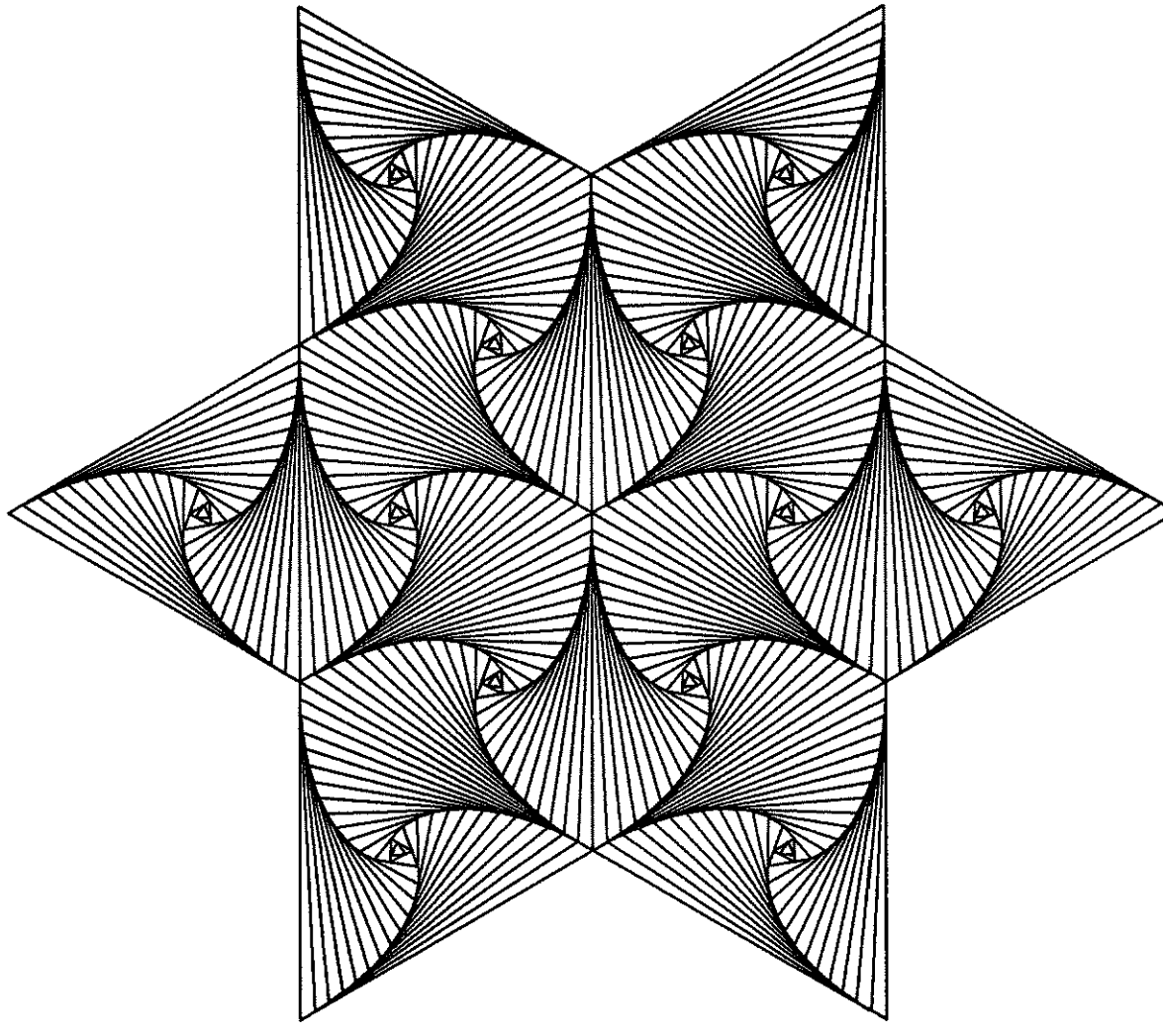
3) *Feitenkennis* over computers.
Hiermee bedoel ik feiten als: een bepaald type computer heeft een kerngeheugen van 32 K woorden van 27 bits, men onderscheidt cilinderdrukkers, kettingdrukkers, letterstaafdrukkers, de draaisnelheid van een schijvengeheugen enz.
Deze leerstof is natuurlijk gegarandeerd verouderd wanneer Uw leerling de school verlaat. Deze mogelijkheid noem ik hier speciaal omdat, tot schande van het onderwijs binnen het koninkrijk, de eerste poging tot het afnemen van een eindexamen in computerkunde juist dit soort leerstof betrof.

4) *Voortgezet programmeren.*
De behandeling van subroutines, recursiviteit, programmeren met een stapel, datastructuren, enz.
Hiermee is geen gemakkelijke maar zinvolle vulling gevonden, met vele mogelijke toepassingen. Handel en industrie tonen belangstelling voor een dergelijke opleiding, om de duidelijke reden dat men op de programmeursopleiding hoopt te besparen. Deze cursus voorziet in een gedeeltelijke beroepsopleiding voor programmeur en dat is alleen voor middelbaar en hoger beroepsonderwijs te verdedigen.

5) *Toepassingen uit andere leervakken*
Het is heel goed denkbaar dat de docent computerkunde toepassingen behandelt van computergebruik in andere leervakken, zoals wiskunde (inklusief waarschijnlijkheidsrekening en statistiek), natuurkunde, biologie, economie, boekhouden, aardrijkskunde, literatuuronderzoek.
Hierbij rijzen twee bezwaren. Welke leraar computerkunde, hoe breed zijn achtergrond ook moge wezen, zal zich thuis voelen in al deze disciplines, om hierover gefundeerd te kunnen spreken? Verder zal Uw gehoor bestaan uit een groep leerlingen met totaal verschillende keuzepakketten. Snijdt U een natuurkundig onderwerp aan, dat zullen de leerlingen die dit vak niet gekozen hebben,weinig begrip en belangstelling tonen. Zo zal elk onderwerp dat U kiest maar een gedeelte van Uw groep kunnen boeien. En ik weet uit eigen ervaring, dat dit zacht is uitgedrukt.

6) *Computerkunde toegepast in andere leervakken*
Het voorgaande punt bevat wel de sleutel tot de oplossing die ik zou willen aanbevelen. Als een computertoepassing in bijvoorbeeld de natuurkunde behandeld wordt in de natuurkundeles, dan is het natuurkundige aspect zeker duidelijk aan de toehoorders of kan duidelijk gemaakt worden door de terzake kundige docent. U werpt nu twee bezwaren op.
Ten eerste: zijn alle leerlingen in de natuurkundeles op de hoogte van programmeren? Als antwoord hierop moet misschien nog 'nee' gelden, maar 'ja' is toch wel het ideaal, zoals U kunt lezen aan het eind van punt 5 van § 1.1. Ten tweede: is de leraar natuurkunde (of welk vak ook) op de hoogte van de mogelijkheden die de computer biedt voor zijn vak? Veel recentelijk afgestudeerde collega's zijn dat wel, maar in de meeste gevallen zullen we ook hierop ontkennend moeten antwoorden.

Als U na lezing van het voorgaande overtuigd bent, dat facet 6 de ideale situatie aangeeft, probeert U Uw collega's dan te wijzen (en warm te krijgen voor) de mogelijkheden, die de computer ook in hun vak biedt.



1.3 INTERNATIONALE AANBEVELING

Van de 'Organisation for Economic Co-operation and Development' (OECD) is in 1971 een aanbeveling uitgegaan naar alle regeringen,met richtlijnen voor een inleidende cursus computerkunde. Deze richtlijnen zijn opgesteld door een werkgroep, waarin van de 42 lidlanden er 8 (waaronder Nederland) vertegenwoordigd waren. Alleen een gedeelte van wat handelt over de minimale inhoud van de cursus is hieronder overgenomen. Aan Module 2, II (a)(3) is voor de duidelijkheid toegevoegd het hier ten lande meer bekende SERA.

Module 1: Automata, Algorithms, Flowcharts

Introduction to automata: i.e. devices that are able to perform a few well-defined operations. The order in which the operations are performed is vital. Examples: juke-box, washing machine, control unit of an elevator, knitting (human automaton). Also of importance is the fact that instructions to an automaton, as distinct from human beings, should be unambiguous and complete. The fundamental idea of an algorithm to be used in this course is simply an ordered set of instructions. Algorithms can be described by flowcharts and consequently some skill must be developed in flowcharting. It is a waste of time to give too much attention to linear flowcharts. From the early stages branching and looping should be used.

Examples of situations which give rise to interesting algorithms, and are suited te be followed-up with simple exercises are:

a) Human activities as putting stamps on letters according to value of stamp and destination of letter, two people playing a hazard game for money, changing wheels of a car from front to rear with a jack and a spare wheel, etc.
b) Administration (payroll)
c) Booking policies (e.g. airplanes)
d) Inventory policies (e.g. supermarket ordering goods)
e) Sorting
f) Selection of library cards
g) Gregorian calendar
h) Arithmetical operations that are constructed from simpler ones.

The chosen examples should be mostly of a non-mathematical nature. The instructions which are used should be on a verbal level of the type: 'take next card, put on a 25c stamp, is the light off'? etc... They will generally differ from problem to problem, and it is advisable to provide the pupils with the standard instructions to use. (if they must find them themselves pupils generally tend to expand into irrelevancies).

It should be pointed out that carrying out a given algorithm is just routine work, very much suited to an accurate but stupid automaton, whereas devising algorithms is by no means routine - it is a highly creative job.

It could be tempting in this module to treat sophisticated algorithms (each of the examples a-f could give rise to that) but this should be resisted. Such examples can be taken up in the second set of modules on applications, after some programming skill has been acquired and they will thus function in a proper setting.

On the other hand some of these examples could be used to illustrate the operations of a computer. This would require that the examples be translated into numerical terms and that the pupils would need to use the operations of addition, subtraction, comparing, etc. and in fact move closer to the instructions used in a computer.

Module 2: Programming

I. Introduction

In this module the pupil will make contact with the computer and will learn the language which allows communication with it. In consequence it is important that, particularly in the early stages the pupil is given a general understanding of the structure of the computer and its different parts (central processor, memory, input, output devices, control unit). This should arise naturally when programming is introduced and indeed will be essential for the proper understanding of the various operations. Programming is only one step in the solution of a problem. The complete sequence of events is:

1) Analysis of problem
2) Preparation of flowcharts
3) Preparation of programme
4) Data preparation
5) Testing and running of programme
6) Results

The most important stages in the solution of a problem are the first two which:

a) define the problem
b) define the solution.

Programming merely expresses the solution in a form suitable for computer interpretation. Pupils should therefore be ancouraged to draw flowcharts which define the solution to the problem and using these as precise plans hence produce the set of programma instructions. This approach ensures that the pupil when writing a programme is concentrating solely on the problems of language and is not side-tracked by the logic of the solution.
Some experience of programming is essential although it is not necessary to spend a great deal of time on it. Neither is it necessary for all pupils to write more than a very few simple programmes.

II. Programming Languages

There are a large number of programming languages available but it should be noted that not all languages are available on all computers. Consequently, the choice of language must depend largely on what facilities are available since no language however desirable should be used unless the programmes written by the pupils can be run. It should be noted that whichever existing language is used there will be undesirable teaching features.

There are two main types of programming languages:

a) Machine Dependent Languages
These are programming languages the nature of which is determined by the structure of a particular computer.

1 - Machine Code
2 - Assembly Language
3 - Language representing an imaginary machine (e.g. MIX, SERA)

Because of the unwieldiness and the technical and machine dependent nature of these languages, it is not recommended that they be used in this course for programming applications.

b) High Level Languages
Such languages are designed to cope with a specific range of applications rather than for a particular computer. A programme written in a high level language can (usually with minor modifications) be run on different computers.

If a teacher is offered facilities using a high level language, it is hoped that he will use the language very simply and not take advantage too early in the course of all the hig level features.

III. Development

Irrespective of which high level programming languages, or subset of a high level programming language is used the development in each case should be gradual with the introduction of one language feature at a time.

Stage (i): Simple programmes to add, subtract, multiply and divide introducing ideas of assignme input, output, comparing, branching, counting.

Use should be made of some of the examples of Mod le 1 but they must be kept simple.

At this point, a minimum amount of programming ha been covered so Module 3 could follow but teacher who choose to end their work on programming here will find that they are limited to a small range fairly trivial examples when they work Module 3.

Stage (ii): (if desirable).
More branching, cycling, one dimensional arrays, files.

Module 3: Computers and Society

This section of the work provides the raison d'êt for the whole of the introductory course. As has already been pointed out, many aspects of the use of computers in society will have been used to mo tivate the earlier work on programming, but some facility in communicating with the computer havir been established, it should now be possible to

discuss with greater understanding the various implications of computer use. These implications - political, social, ethical, economic and legal - must where appropriate be discussed together with questions such as 'who takes decisions?'. In this context pupils must be led to appreciate that the computer on its own does not decide but rather that the decisions are still human responsibilities and that these are taken at various levels by programmers, systems analysts, administrators, etc.

It is impossible to provide an exhaustive list of computer uses. Immediately obvious are such applications as payrolling, accounting, billing and so on, and indeed pupils may have already treated these at an elementary level in their work on programming. From these types of situations, the pupils should be led to various aspects of computer control. These can come from situations such as: traffic control, that is traffic lights under computer control; credit control, that is the movement towards a society where credit cards will replace money; pollution control, now practised in many industrialised countries; production control, such as steel rolling; and finally the control which could be exercised by the use (or misuse) of data banks.

Pupils should be required to make a case study at some depth on one such topic. Topics which are not covered at depth can be used for discussion with pupils or can be the motivation for visits to various computer users. In addition to the applications already suggested, it could be fruitful for pupils to look at possible future applications. This could include the use of the computer in the fine arts, and could also include some introduction to game playing on the computer.

The work in this module would benefit greatly from the use of a variety of supporting materials. These would include books, wall charts, slides, film loops, and films. Some indication of appropriate material will be given in a document to be publised separately.

2 Aanwijzingen voor de les

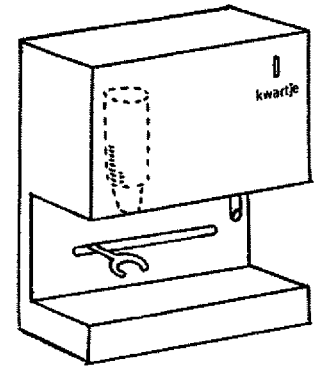
2.1 DE LEERSTOF

2.1.1 Het blokschema (Compk § 3.1, Werks hfdst. 1)

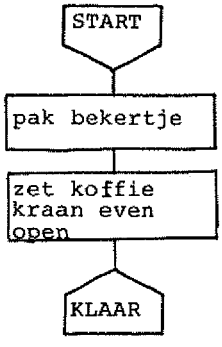
Waarom moeten blokschema's behandeld worden? Men hoort vaak protesten dat men de leerling de programmeertaal kan bijbrengen zonder over blokschema's te spreken. Dat is juist. Maar men bedenke dat in onze doelstelling niet uitsluitend het leren programmeren staat vermeld. Belangrijker is het leren analyseren van een probleemstelling. En daarvoor is het blokschema een aangewezen, hoewel mogelijk niet het enige, expressiemiddel. Voor U als docent is verder van belang dat U in Uw uitleg altijd terug kunt vallen op het uitbeelden in blokschema's. Een leerling die bij het maken van een wat ingewikkelder programma steeds fouten maakt kan, zonder hulp van blokschema's, haast niet op het juiste spoor gezet worden.

Het blokschema wordt gepresenteerd als middel om een proces te beschrijven. Met opzet zijn in de voorbeelden en opgaven processen gekozen uit het dagelijks leven. De overgang voor de leerlingen naar dit nieuwe leervak gaat daardoor zonder drempel en het werkt animerend om in de wereld om ons heen processen te herkennen. Een eenduidig voorschrift om een proces uit te voeren met eindig veel handelingen uit een repertoire van welbegrepen en uitvoerbare handelingen, wordt een *algoritme* genoemd.
Geleidelijk kan men overgaan op algoritmen voor een automaat. Als voorbeeld hiervan het volgende klassegesprek (naar aanleiding van Compk opgave 3.1).

docent
We stellen ons een koffie-automaat voor. Er is een gleuf voor kwartjes. In het inwendige is een stapel bekertjes. Een klauwtje kan een bekertje onder een tuit houden. Uit deze tuit kan bruin vocht stromen, dat goedwillende mensen ook wel koffie noemen. Schrijf een blokschema dat de werking van de automaat aangeeft nadat er een kwartje is ingeworpen.

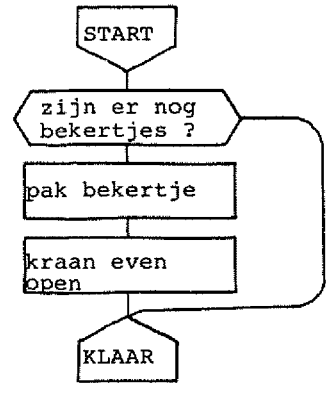


leerling

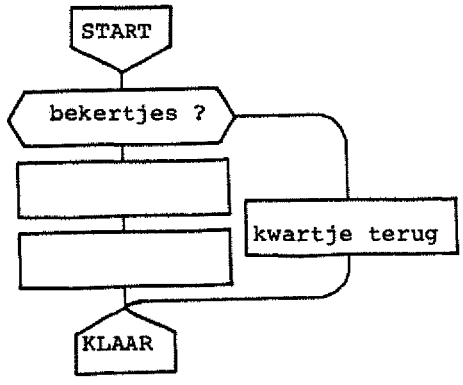


Dit betekent: er valt één beker van een stapel in een klauwtje. Het klauwtje schuift onder de tap.
Heet water stroomt net lang genoeg door oploskoffie om een beker te vullen

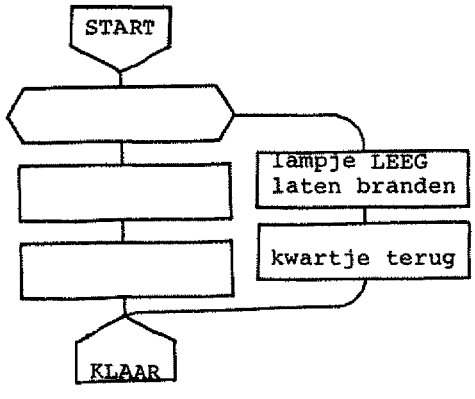
docent Wat kan er fout gaan ?
leerling De stroom kan uitvallen.
docent
(Realiseert dat zijn vraag te vaag was). We gaan een blokschema maken voor wat er *in* de automaat gebeuren moet. Moeilijkheden *buiten* de kast kunnen we niet verhelpen. We nemen nu maar aan dat er altijd electriciteit en water is. Wat kan er fout gaan als de automaat stipt dit schema opvolgt.
klas
Het kan voorkomen dat er geen bekertjes meer zijn.
docent
Verbeter het blokschema met de vraag 'Zijn er nog bekertjes?'.
leerling

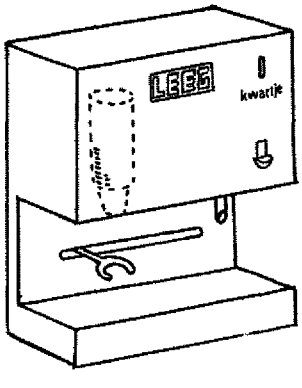


docent
Heeft iemand bezwaar tegen een automaat die werkt volgens dit schema ?
leerling
Als de bekertjes op zijn, ben je je geld kwijt zonder iets te krijgen. Het schema moet zijn:



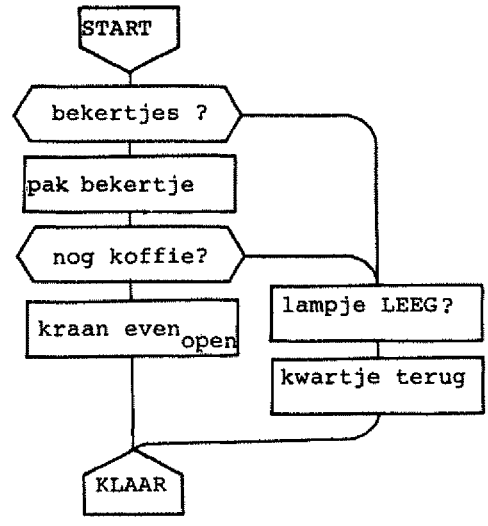
docent
Deze automaat is wel onduidelijk. Wanneer er geen bekertjes meer zijn, komt het kwartje onmiddellijk terug. De klant zal het dan nog eens met een ander kwartje proberen. Steeds zonder resultaat. We zullen daarom de automaat uitbreiden met een lampje met 'LEEG' er op. Geef dit in het blokschema aan.
leerling



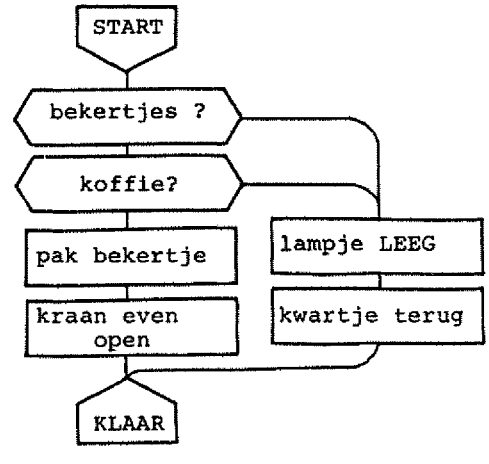


docent
Is er nog een probleem waarin ons blokschema niet voorziet?
klas De koffie kan ook nog op zijn.

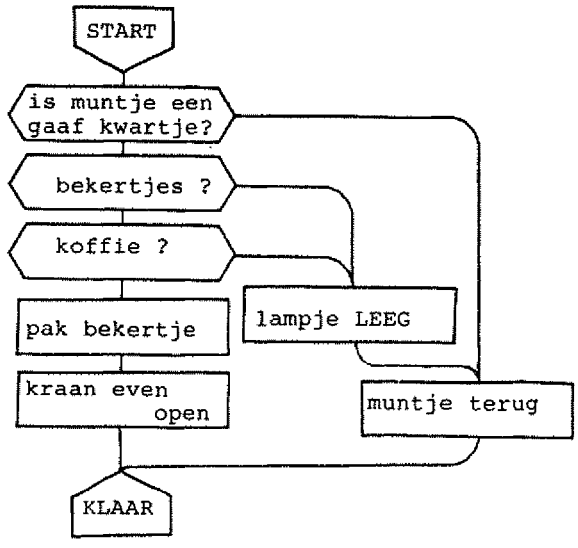
docent
Ik zal het schema veranderen.



Is het zo goed ?
klas (5 minuten verbaasde stilte)
leerling
Als er wèl bekertjes zijn en geen koffie en iemand gooit steeds opnieuw zijn kwartje er in, dan valt er steeds een nieuw bekertje in het klauwtje. Op den duur zal de boel klem lopen door de stapel bekertjes.
docent
Goed zo. We moeten er nu eenmaal rekening mee houden, dat de automaat geen verstand heeft. Dus moeten wij ons verstand wel gebruiken om het blokschema ook in deze gekke situatie te laten voorzien. Verbeter het blokschema.
leerling



docent
Zijn er nog meer voorzieningen die we willen treffen ?
klas Moet het kwartje nog gekontroleerd worden?
docent Probeer maar.
klas (na enig puzzelen)

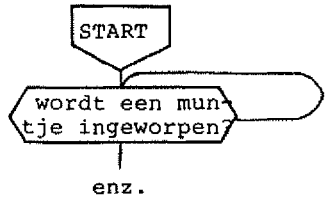


kommentaar

In een klassegesprek is het mogelijk om tijdens de oplossing af te spreken, welke vragen of opdrachten nog wèl en welke niet 'verstaanbaar' zijn voor de automaat. Een opgave waarbij de leerling zelfstandig een blokschema moet opstellen, mag nooit zo vaag gesteld zijn als bijvoorbeeld: 'maak een blokschema voor een koffieautomaat'. Een leerling die als oplossing geeft:

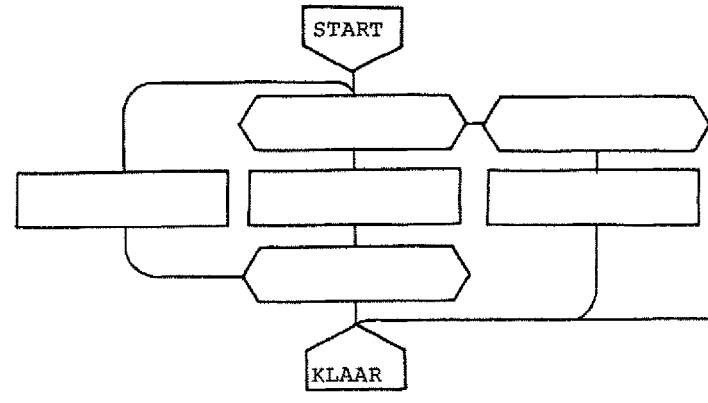
heeft dan in wezen gelijk. De kans is echter nog groter dat de leerlingen zich uitputten in de kleinste detailhandelingen. De leerlingen moeten bij de opgave dus altijd voorzien worden van een verzameling van elementaire vragen en opdrachten die voor de automaat (in ruime zin) nog net verstaanbaar zijn (zogenaamde '*primitiva*'). Vergeet U bij Uw blokschemavraagstukken ook nooit de verzameling primitiva te geven.

De besproken opgave luidde met opzet: de werking *nadat* een kwartje (of muntje) is ingeworpen. U kunt zich beter niet inlaten met blokschema's als:

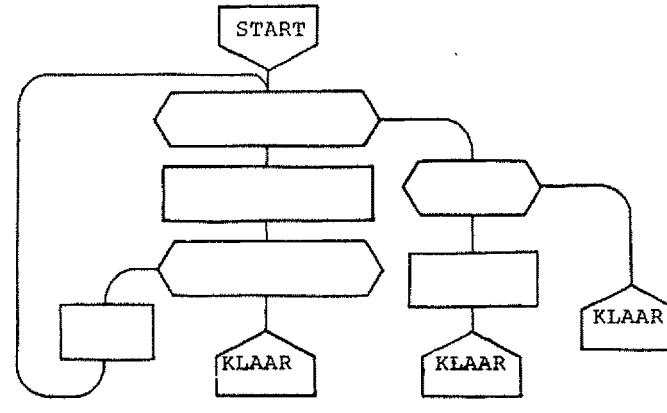


Dit is een cyclus waarbij we moeten veronderstellen, dat het beantwoorden van de vraag tijd kost. Maar hoeveel tijd?

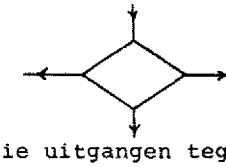
Men lette speciaal op het naleven van conventies bij deze blokschema's. De 'ingang' van elk blok is midden boven. Let op beide, veel voorkomende, fouten in onderstaand schema.



Ook zal men slechts één KLAAR-blok per schema toestaan. Dus niet dit:



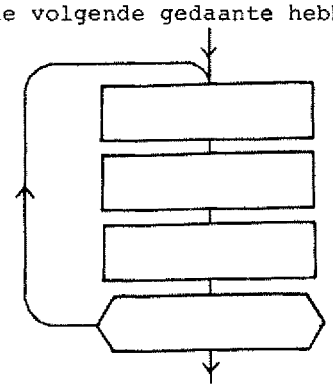
Deze conventie wijkt af van de inmiddels in Nederland genormaliseerde vorm. De laatste beveelt bijvoorbeeld voor het vraagblok aan:



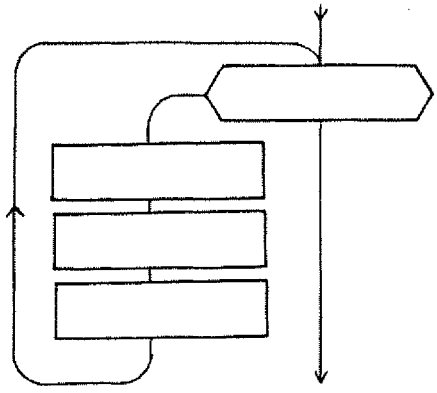
waarbij drie uitgangen tegelijk gebruikt mogen worden, zonder dat de betekenis vastligt. De ruit heeft het bezwaar gauw groot genomen te moeten worden bij veel tekst. Het twee uitgangenprincipe in onze leerboeken werd verkozen in verband met de programmeertaal. Dat de ja-uitgang altijd op dezelfde plaats (midden onder) zit is geen enggeestigheid, maar werkt op den duur alleen maar prettiger. Zou dat niet zijn afgesproken, dan moet bij elke uitgang van een vraagblok 'ja' of 'nee' worden bijgeschreven. Iets wat leerlingen al gauw verwaarlozen.

2.1.2 Blokschema's met herhalingen (Compk § 3.2, Wrks hfdst. 6)

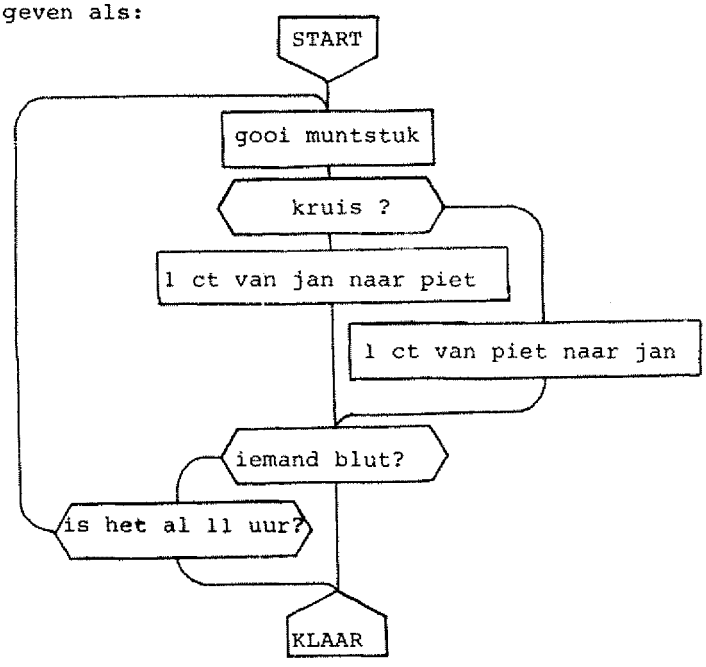
In de tekst wordt gewezen op de noodzakelijkheid van een vraagblok (splitsing) in een cyclus. Uw leerlingen zullen vanaf dit moment voortdurend gewezen moeten worden op het gevaar van een eindeloze cyclus. Blokschematisch zal een goede cyclus iets van de volgende gedaante hebben.



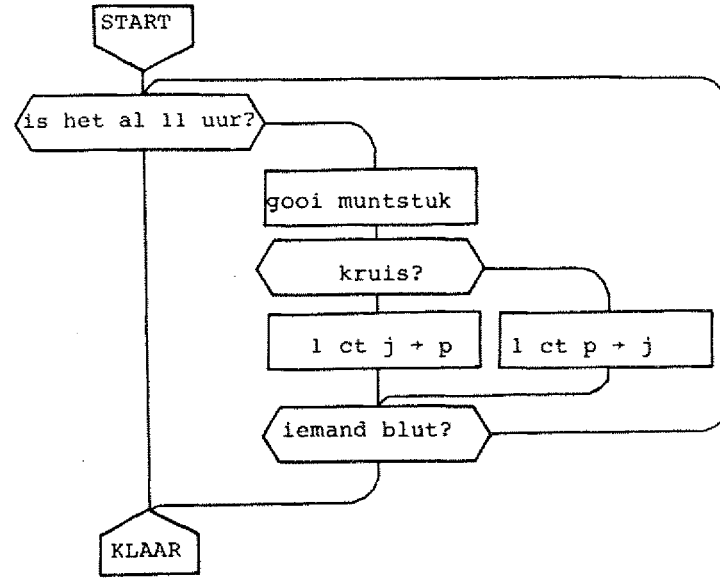
In veel gevallen is het raadzaam om het vraagblok 'eerder' in het schema op te nemen, bijvoorbeeld:



De oplossing van Compk opgave 3.9 wordt vaak gegeven als:



Als U echter bedenkt dat Jan en Piet wel eens na 11 uur het idee kunnen krijgen om te gaan gokken, dan zou U dat waarschijnlijk willen verbieden. Volgens bovenstaand blokschema wordt er toch één maal kruis-of-munt gegooid. Dit wordt voorkomen in onderstaand schema.



Ook de blutvraag zou boven het muntstuk gooien geplaatst kunnen worden, maar dat is minder urgent zolang bekend is dat beide gokkers met 10 ct beginnen.

2.1.3. Programmeren

Na het eerste hoofdstuk wordt een begin gemaakt met het programmeren. Een programma is een algoritme, geschreven in een vorm die, na eventuele tussenbewerkingen, geschikt is voor een computer. Met de tussenbewerkingen wordt bedoeld: het verponsen op kaarten of papierband of het aanschrappen op kaarten.

Waarom programma's schrijven? Is het niet voldoende dat door een leerling is geanalyseerd en dat hij de oplossing in blokschema heeft aangegeven? Daartegen kunnen drie argumenten worden ingebracht.

1) De leerling moet kennismaken met de hele weg die een probleem aflegt om op een computer verwerkt te worden, zodat rondom de computer zo weinig mogelijk geheimzinnigheid blijft bestaan.
2) Het is voor leerlingen bijzonder stimulerend om ook resulaten te krijgen uit de computer, na al het voorbereidende werk wat hij er aan heeft verricht.
3) Door de computer wordt er de nodige kontrole op het werk verricht. Hij krijgt de ervaring dat een vrijwel foutloos algoritme bijzonder storende fouten in de resultaten kan veroorzaken. Argumenten als 'het komt uit de computer, dus is het zo' zijn hem hierna vreemd.

2.1.4 Onvertakte rekenschema's (Compk hfdst. 5, Wrks hfdst. 2)

De blokschema's in het bedoelde hoofdstuk zijn zo eenvoudig, dat ze niet eenvoudiger of prettiger leesbaar zijn dan de overeenkomstige programma's. De illustrerende waarde van blokschema's gaat pas waarde krijgen, wanneer vertakkingen optreden. Toch is hier het blokschema gehandhaafd uit continuïteitsoverwegingen.
Dit hoofdstuk is een eerste kennismaking met een vast handelingenrepertoire, in tegenstelling tot het voorgaande hoofdstuk, waarin in elke opgave de toegestane verzameling (vragen en) opdrachten anders was.

Geheugen
Men wijze de leerlingen erop dat beslist niet het menselijk begrip geheugen bedoeld wordt. Het computergeheugen is uitsluitend geschikt voor bewaren van getallen (ook al kunnen deze als opdrachten geïnterpreteerd worden). In het menselijk geheugen wordt informatie teruggevonden door associatie, in het computergeheugen door plaatsbepaling. Deze geheugenplaatsnummering of adressering wordt in de programmeertaal vervangen door het gebruik van geheugenplaatsnamen.

In andere programmeertalen is er soms een andere schrijfwijze voor het wordt-teken, zoals "=" (kan verwarring wekken met het algebraïsch gelijkteken, zoals in a = a+2) of "←".
Het laatste teken geeft wel duidelijk aan dat het rechter lid eerst berekend wordt en daarna het resultaat geplaatst in de geheugenplaats die links wordt aangegeven.

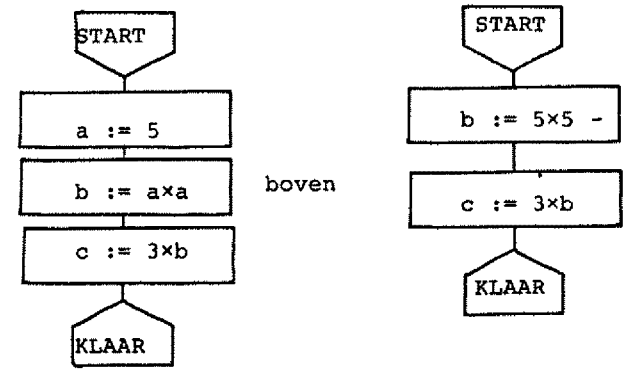
Punten die verder de nadruk verdienen zijn:
- het lezen van een getal uit een geheugenplaats houdt niet in dat het getal daarna van die plaats verdwenen is.
  b := a is een opdracht om te kopieëren, niet om te verplaatsen. De zinsnede 'haal een getal uit een geheugenplaats' is misleidend.

- een geheugenplaats is nooit leeg. Men mag er niet op rekenen dat een in het programma nog niet gevulde geheugenplaats automatisch een nul zou bevatten.

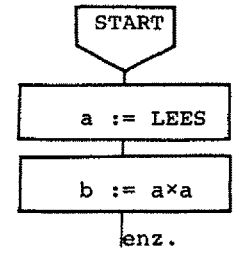
- het werken met geheugenschema's (of kaartenbakgeheugens) moet worden gestimuleerd .

Als het tijdens Compk § 5.3 of Wrks hfdst. 2 te pas komt, kan de schrijfopdracht al wel eerder worden ingevoerd.

Het is mogelijk raadzaam een (niet tegenover de leerlingen uitgesproken) voorkeur te hebben voor:



voor het berekenen van $3a^2$. Het linker blokschema geeft namelijk later een betere overgang naar:



2.1.5 Dezelfde geheugenplaat aan beide zijden van het wordt-teken (Compk 5.4, Wrks vanaf 7.6)

Voor een juist begrip voor dit soort opdrachten is het oefenen met geheugenschema's voor de leerlingen onontbeerlijk.

In Compk 5.4 wordt gesproken over het bezuinigen op het aantal geheugenplaatsen. Het is niet raadzaam om het bezuinigen op het aantal geheugenplaatsen in willekeurige programma's aan te moedigen. De kans op programmafouten neemt alleen maar toe en de leesbaarheid van het programma af.

Naar aanleiding van Compk opgave 5.16 het volgende.
De werking van de blokschema's moeten worden gekontroleerd. Laat U dat de leerlingen doen met twee getalwaarden voor a en b. Bijvoorbeeld 20 en 3. Voor Uw gevoel zal dit misschien weinig bewijskrachtig zijn. Maar het is beslist te ontraden om een bewijs te leveren door in

a := a+b
b := a-b
a := a-b

de begingetallen voor te stellen door bijvoorbeeld p en q.

a	b
p	q
p+q	q
p+q	(p+q) - q = p
(p+q) - p = q	p

De leerling zal de rest van de cursus het idee houden dat er op deze wijze letters kunnen worden opgeborgen in geheugenplaatsen.

Als oplossing van Compk opgaven 5.17 en 5.18 kunt U drie verschillende beelden verwachten:
- de zelf gekozen getallen komen door de hele berekening voor. Een op zichzelf juiste oplossing, maar niet fijn voor de docent om mee te werken;
- een berekening met variabelen waaraan geen waarden zijn toegekend (wordt dit gesuggereerd door BS 5.14 t/m 5.17 ?). Een foute oplossing;
- eerst een toekenning van waarden aan de variabelen, daarna een berekening met die variabelen.

De laatste oplossing verdient de voorkeur, i.v.m. de latere leesopdracht. De eerste oplossing kan mogelijk voorkomen worden door een aanwijzing van de docent.

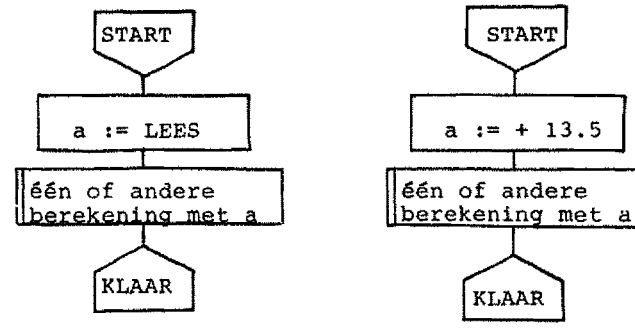
2.1.6 Schrijven en lezen
(Compk hfdst 6, Wrks hfdst 3 en 9)

De schrijfopdracht geeft in het algemeen weinig problemen. Op het moment dat de programma's inderdaad verwerkt worden, kan nog enige aandacht worden geschonken aan het preciese effect van de opdracht SCHRIJF := a
zoals: hoeveel cijfers voor en achter de decimale punt, wanneer wel en wanneer geen voorteken, welke notatie bij een getal van een groot aantal cijfers voor de punt.

Het verwisselen van linker en rechter lid bij de lees- en de schrijfopdracht komt wel voor.

Toch is het principe gehandhaafd, dat rechts van het wordt-teken de bron staat en links het doel.

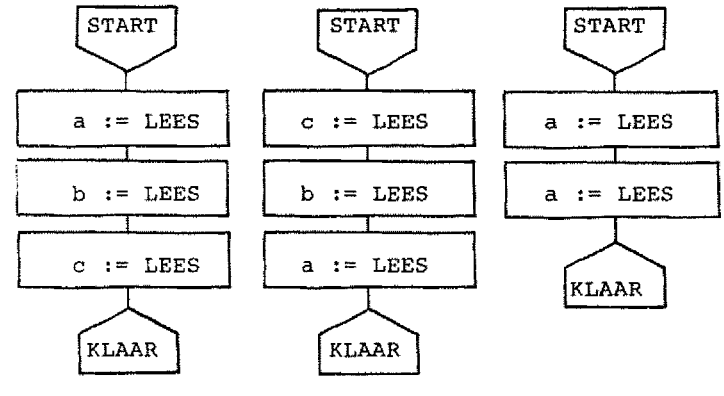
De leesopdracht.



getalkaart + 13.5

Het voordeel van het linker blokschema boven het rechter is aan leerlingen niet zondermeer duidelijk. Dit voordeel moet verdedigd worden met het feit, dat het linker programma algemener bruikbaar is, d.w.z. dat de berekening uitgevoerd zou kunnen worden met verschillende waarden voor a. Dit wordt echter pas realiteit als het programma herhaald wordt door middel van een cyclus.
Men lette overigens op het verschil tussen algebra en een computerprogramma. In de algebra wordt een berekening met letters uitgevoerd, zodat achteraf substitutie van waarden kan plaatsvinden, in een computerprogramma vindt eerst 'substitutie' plaats d.m.v. de leesopdracht en daarna de berekening met namen.
De nadruk moet gelegd worden op het feit, dat na KLAAR getallen genoemd worden *zonder* namen van geheugenplaatsen. Het blokschema moet zorgen dat deze getallen de juiste geheugenplaatsen gaan bezetten.

Oefening met eenvoudige gevallen is vaak wenselijk, zoals: welk getal zit er in geheugenplaats a in elk van de volgende gevallen?



+ 15 + 13 + 12 + 15 + 13 + 12 + 15 + 13 + 12

2.1.7. De eerste programma's
(Compk hfdst 7, Wrks hfdst 5)

Deze leerstof bevat de volgende nieuwe elementen:
- overgang van blokschema naar programma
- de uitvoeropdrachten TEKST en NR (dit geldt niet voor Wrks, waarin deze opdrachten in hfdst 4 behandeld worden)
- het opnemen van naam en klas in het programma
- de plaats van de te lezen getallen (geldt niet voor Wrks)
- het omzetten van het programma in een voor de computer acceptabele vorm zoals de schrapkaart).

Het is af te raden dit alles in één les te behandelen. Verdeling over meerdere uren zal wel met zich meebrengen, dat meer oefenstof zal moeten worden bedacht.
Aanwijzingen over het gebruik van de schrapkaarten kunt U vinden in §6.1. Bij eenvoudige (onvertakte) programma's lijkt een blokschema overbodig. Toch is het een goede gewoonte om elke verwerking door een computer in zes stappen te nemen:
- blokschema
- programma, geschreven op papier
- programma, geschreven op de schrapkaarten
- opdrachten aanschrappen op de schrapkaarten
- verzending
- bij terugontvangst, controle van de resultaten.

De eerste programma's dienen vooral om de theorie over geheugenplaatsen aan de werkelijkheid te toetsen.
Voorbeeld:

```
START
TEKST := "j.f,3a"
NR
a := 1
b := 3
SCHRIJF := a
SCHRIJF := b
NR
c := a
a := b
b := c
SCHRIJF := a
SCHRIJF := b
SCHRIJF := c
KLAAR
```

In sommige computers wordt geaccepteerd dat de opdracht a := b wordt gebruikt, terwijl aan b nog helemaal geen waarde is toegekend. Wel blijkt a dan een volkomen absurde waarde te hebben.

De verklaring is de volgende:
Het ECOL-programma wordt gelezen, bekeken op onbekende hoofdletter-woorden e.d.. Elke onduidelijkheid in de ECOL-tekst geeft aanleiding tot foutmeldingen. De opdracht a := b is op zichzelf niet fout en wordt daarom overgenomen.
Tevens wordt aan elke voorkomende variabele een geheugenplaats toegewezen (voor insiders: real gedeclareerd). Totdat het programma de juiste waarde toekent aan die variabele is de inhoud van zo'n geheugenplaats een onvoorspelbare chaos. Als b in het programma geen waarde heeft gekregen, zal bij de opdracht a := b de toevallig in b aanwezige rommel worden overgedragen aan geheugenplaats a.

Door de meeste docenten wordt de mening gekoesterd, dat juist in de eerste programma's zo weinig mogelijk door de leraar moet worden gecorrigeerd. Laat de leerling ervaren dat de computer een slecht verstaander is, die precies op de juiste wijze moet worden 'aangesproken'.
Foutmeldingen zijn anders het onherroepelijke gevolg. Bij latere meer ingewikkelde programma's kan de docent foutjes tegen de taal vóór het verzenden laten verbeteren. Hebt U hierover een andere mening, dan voegt U zich bij een minderheid, die zeker niet klein is.

Probeert U de leerlingen te overtuigen, dat het meer zin heeft de eerste programma's zo te schrijven dat ze nagerekend kunnen worden. Pas dan kunnen we zeker zijn over de werking van de programma's. Hierdoor wordt voorkomen dat de leerlingen gaan wedijveren in het gebruik van immens grote getallen en enorme aantallen geheugenplaatsen. De resultaten van programma's die hierin uitblinken, hebben voor de leerling toch geen betekenis of controlemogelijkheid.

Programma's genoteerd op schrapkaarten zijn gemakkelijk te verbeteren en opnieuw in te zenden. Voorwaarde is wel dat de leesbare tekst bovenaan de kaarten staat aangegeven, zodat de onjuiste kaarten gemakkelijk kunnen worden teruggevonden en vernieuwd.

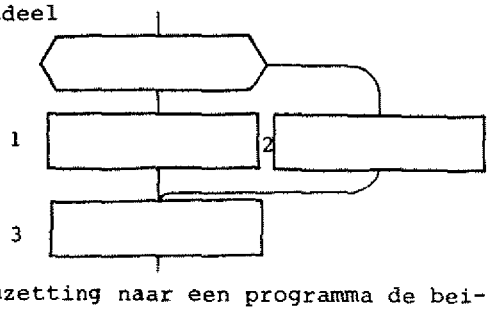
Bij de resultaten krijgt de leerling ook het programma nog afgedrukt, zodat gecontroleerd kan worden of dit via de schrapkaarten juist is doorgekomen.

2.1.8 Vertakking in blokschema (Compk § 8.1, Wrks hfdst 7)

Een veel voorkomende fout is het gebruik van het wordt-teken in plaats van het gelijk-teken in het vraagblok. In Wrks hfdst 7 komt bovendien oefening voor met dezelfde geheugenplaatsnaam aan weerszijde van het wordt-teken en een eerste kennismaking met de tellende cyclus. Elke oefening hierin is broodnodig, alle vraagstukken worden daarom van harte aanbevolen.

2.1.9 De sprongopdrachten (Compk § 8.1, Wrks hfdst 7)

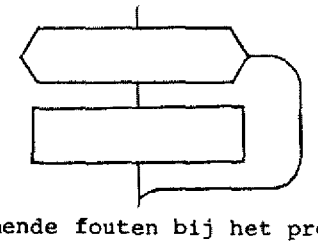
Het blokschemadeel



vereist bij omzetting naar een programma de beide sprongopdrachten:

```
     ALS ..... DAN 1 ANDERS 2 (voorwaardelijke
                              sprong)
1    ..............
     NAAR 3
2    .............
3    .............
```

U kunt ook eerst alleen de voorwaardelijke sprongopdracht introduceren door voorbeelden te zoeken, waarvan de splitsing de volgende gedaante heeft.



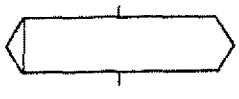
Veel voorkomende fouten bij het programmeren met splitsingen zijn:

- in de sprongopdracht staat een nummer genoemd dat niet als regelnummer voorkomt. De foutmelding verwijst meestal naar het ontbreken van een regelnummer voor KLAAR. Dat is namelijk de laatste regel, waarop het gezochte nummer zou hebben kunnen staan.
- een nummer wordt meerdere malen als regelnummer gebruikt
- een wordt-teken tussen ALS en DAN, in plaats van een gelijk-teken.

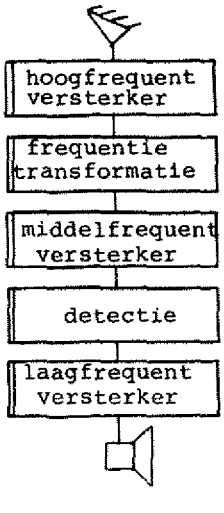
Het grove blokschema (Compk § 8.3, 8.4)

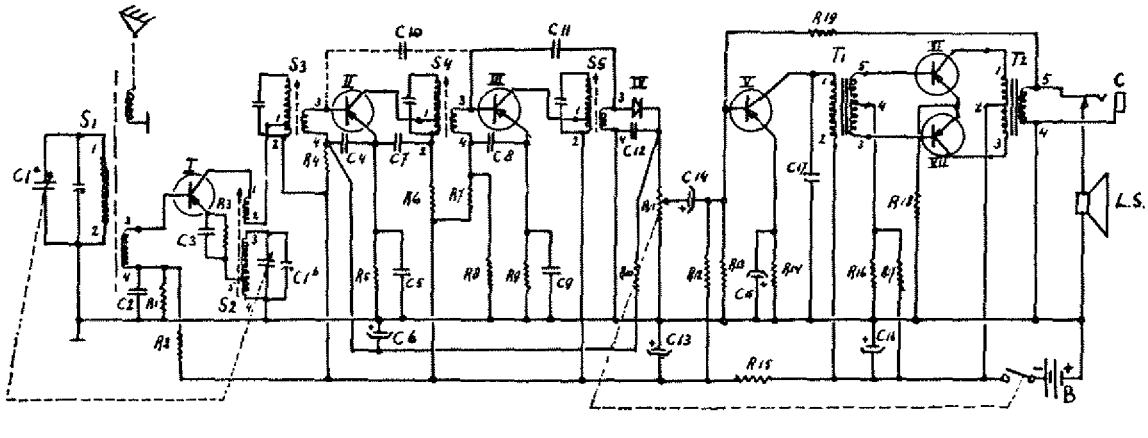
Bij het organiseren van een systeem is het zeer gebruikelijk om eerst een oplossing aan te geven in grote stappen, waarna elke stap in details uitgewerkt kan worden door verschillende mensen of teams. In de bedoelde paragrafen is dat aangegeven door zogenaamde grove blokken. In

wordt een opdracht aangegeven, die niet tot de verzameling van ECOL-opdrachten behoort. Voor de splitsing



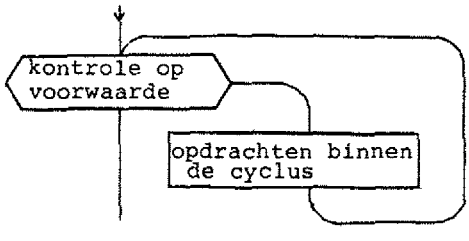
geldt hetzelfde. Dit onderwerp is in kleinere letter en in Wrks helemaal niet opgenomen, omdat in het verleden is gebleken, dat door tijdgebrek het onderwerp vrijwel niet ter sprake kwam. Toch wil ik graag een lans breken voor de behandeling ervan.
Naast tijdgebrek werd ook wel geringe belangstelling van de leerling als argument aangevoerd, maar juist de detaillistische instelling van leerlingen zou hiermee doorbroken kunnen worden. Ook het werken in teamverband kan bevorderd worden, omdat het grove blokschema mogelijkheid schept het probleem te verknippen.
In ieder geval zou ik U als docent willen aanbevelen om grove blokschema's te gebruiken om grote lijnen van oplossingen aan te geven.
Men vergelijke dit bijvoorbeeld met de grove organisatie en een detailschema van een radio.



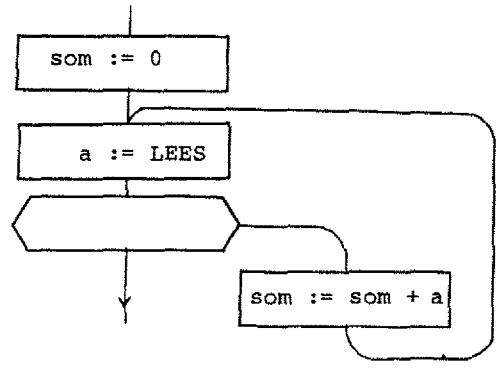


2.1.10 De cyclus

De grote kracht van de computer, het vele malen herhalen van een gedeelte van een rekenproces, waarbij deze herhaling aangegeven kan worden door de automaat terug te voeren naar een punt eerder in het programma. De cyclus maakt het mogelijk veel werk te laten verzetten met weinig programmeren. Onze leerlingen kunnen bij dit onderwerp weer een algoritmische ervaring opdoen: het zoeken naar struktuurdelen die zich herhalen. Overigens moeten sommige leerlingen hiertoe wel worden aangezet. Maar al te vaak verkiezen ze het om een programmadeel vijfmaal uit te schrijven, boven het construeren van een cyclus. Dit laatste vereist namelijk meer denkwerk. De cyclus eindigt op een zekere voorwaarde. Dit kan zijn het ontmoeten van een bijzonder getal op de getalkaarten of het bereiken van een zekere waarde door een variabele. Vergeet men de kontrole op het voldoen aan deze voorwaarde in te bouwen, dan spreken we van een oneindige cyclus. Een oneindige cyclus zou een computer te lang bezet houden, vandaar dat een veiligheid in het systeem is ingebouwd, die de programmaverwerking kan beëindigen na een bepaalde tijd (bijv. 15 sec., afhankelijk van de snelheid van de machine) of na een bepaald aantal sprongen (500 of 1000) veroorzaakt door sprongopdrachten.
Dit afsnijden van de verwerking (de guillotine) kan ten onrechte zijn, in het geval de berekening eindig doch tijdrovend is. In dat geval kan een ruimere tijd worden aangevraagd voor dat speciale programma.
Voor de cyclus geldt (uitgezonderd de tellende cyclus) hetzelfde als wat in 2.1.2 is gezegd, nl. dat een oplossing van de volgende vorm meestal de enig juiste is

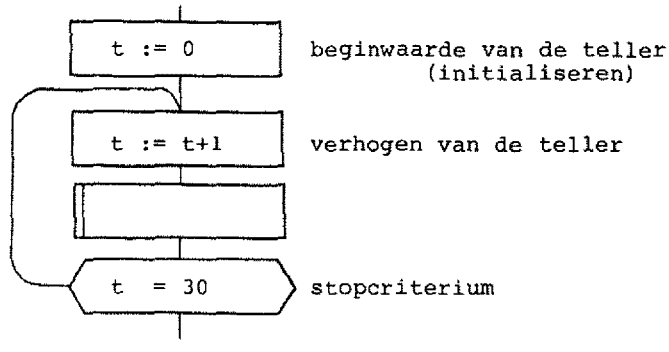


Veel opgaven (Compk 9.1 t/m 9.4, 9.12, 9.17 t/m 9.23, Wrks 9-17, 9-18, 9-19, 10-3, 10-4, 10-5, 10-6) hebben betrekking op een getallenrij, die aan de programmaschrijver onbekend is. De schrapkaarten geven de mogelijkheid om dit realiteit te laten worden, want U kunt een leerling opdragen het programma te schrijven en zijn buurman om de getalkaarten te leveren.



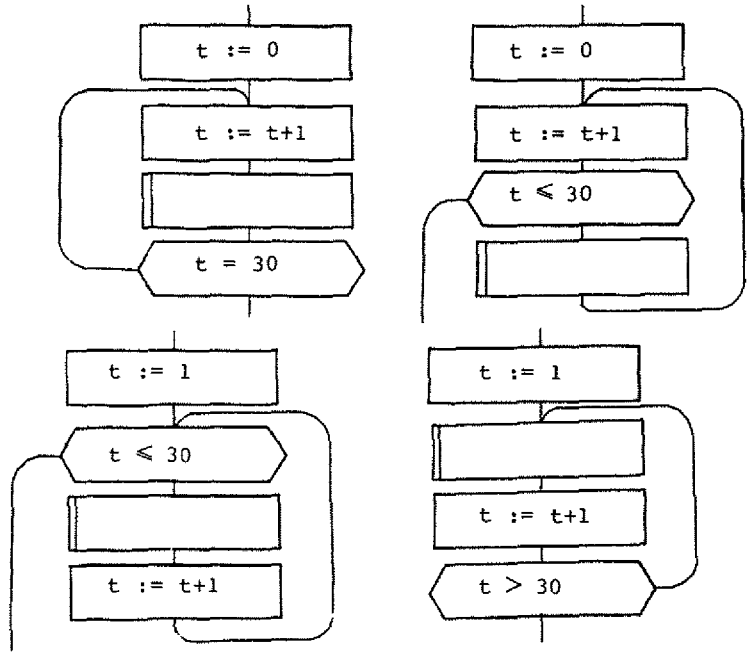
In de constructie, waarin de geheugenplaats "som" als spaarpot fungeert voor het sommeren van een aantal getallen a, moet uitdrukkelijk gewezen worden op de bovenste opdracht (het leeggooien van de spaarpot).

Een bijzonder geval van de cyclus is de tellende cyclus, waarin de voorwaarde voor beëindiging (stopcriterium) gekonstrueerd wordt door een soort telwerk. Hier is te herkennen:



Bij door leerlingen ingezonden programma's mankeert het nogal eens aan één van deze drie noodzakelijke opdrachten. Als docent kunt U vóór het inzenden hier misschien enige controle op uitoefenen.

De tellende cyclus kan op verschillende manieren worden georganiseerd, bijv.:

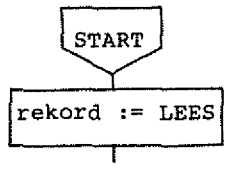


Het verdient de voorkeur om als docent U zoveel mogelijk in de voorbeelden te houden aan één type.

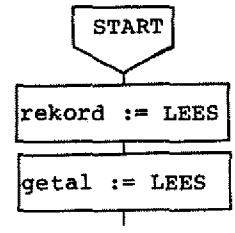
2.1.11 Bepalen van grootste of kleinste getal
(Compk § 9.5, Wrks 9-21, 9-22)

De algoritme voor het bepalen van het grootste (of kleinste) van een aantal op kaarten staande getallen behoort voor veel leerlingen niet tot de gemakkelijkste. Hier volgt een verslag van een VWO-klassegesprek, zodat U zich alvast op het onderwerp kunt oriënteren.

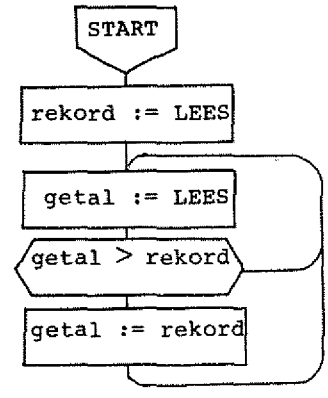
docent
+ 76 + 78 + 77 + 79 + ... + 76 + 0.
Dit stelt een serie resultaten voor van een wedstrijd speerwerpen, behalve natuurlijk de afsluiter nul. Het grootste getal uit deze rij zullen we rekord noemen. Denk je in dat de rij zo'n vijftig getallen lang is. Breng eens onder woorden hoe je zelf het rekord zou opzoeken.
leerling
Je kijkt of je een getal groter dan 79 tegenkomt.
docent
Ja, wij mensen kunnen in één oogopslag het grootste van de eerste vier getallen zien. Dit grootste getal onthouden we en we gaan de rij langs om een groter getal te zoeken. Stel dat we het getal 80 tegenkomen, wat dan?
leerling
Dan gaan we kijken of we een getal groter dan 80 zien.
docent
Is het getal 79 nog belangrijk?
klas
nee.
docent
Wanneer zou je zeker weten dat 80 het rekord is?
leerling
Als je bij de afsluiter komt zonder een getal groter dan 80 gezien te hebben.
docent
Juist. We gaan eens kijken of we dit werkje door een computer kunnen laten doen. De computer kan niet in één oogopslag vier getallen overzien. We lezen één getal



leerling
Maar het eerste getal is helemaal het rekord niet.
docent
In ons voorbeeld niet, maar kun je ook een geval bedenken waarbij dat wèl het geval is?
leerling
Als de volgende getallen toevallig allemaal kleiner zijn dan het eerste getal.
docent
Zolang we nog niet verder gekeken hebben, nemen we aan dat het eerste getal het grootste is. We gaan nu het volgende getal uit de rij lezen:

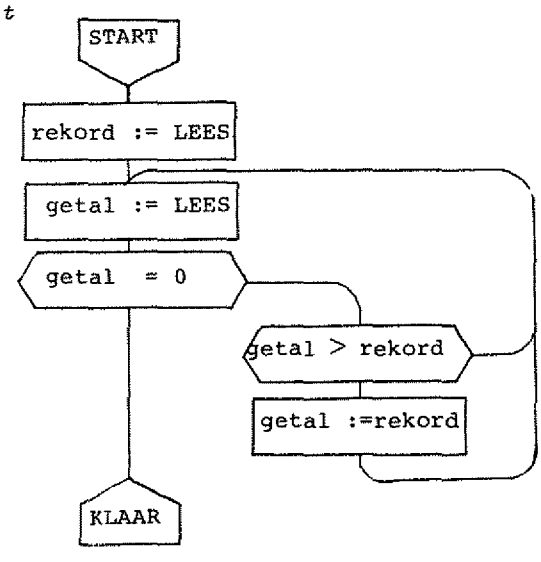


Wat doen we met dit getal?
leerling
Kijken of dit groter is dan het rekord wat we al hebben en als dat zo is, wordt dat het nieuwe rekord.
docent
Goed. Ik zal nu een blokschema tekenen, waarin een aantal fouten zitten. Aan jullie om ze eruit te halen.



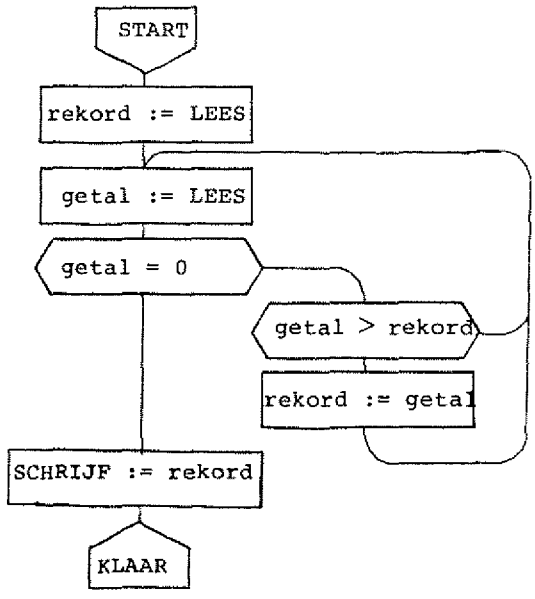
fout blok-schema

leerling
Het is een eindeloze cyclus. We moeten nog kijken of we de nul tegenkomen.
docent



fout blok-schema

leerling
Er moet nog een schrijfopdracht in.
leerling
Dat getal wordt rekord moet toch andersom?
docent



+ 76 + 78 + 77 + 79 + 77 + 80 + 81 + 76 + 0

Jan, maak op het bord eens een geheugenschema om dit blokschema met deze getallen te kontroleren.

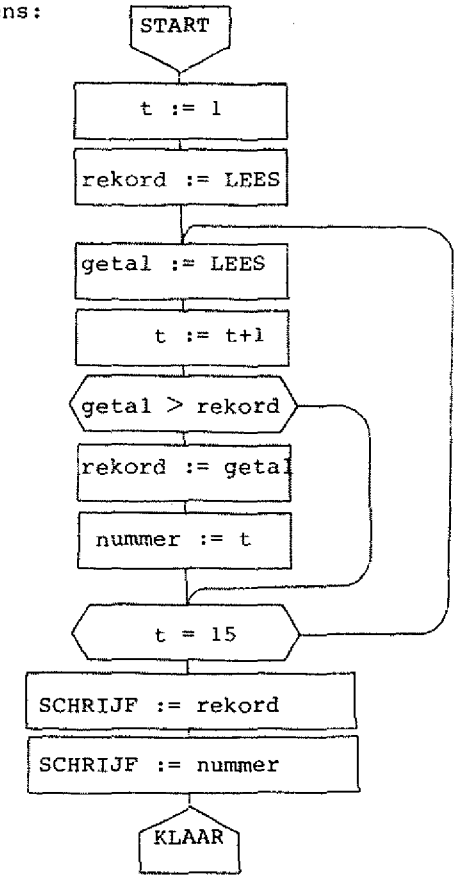
Jan

rekord	getal
+ 76	
+ 76	+ 78
+ 78	+ 78
+ 78	+ 77
+ 78	+ 79

enz.

docent
Je kunt je voorstellen dat bij wedstrijden alle uitslagen door een computer verwerkt worden en dat de wedstrijd speerwerpen is afgelast. Op de getalkaarten staat dan alleen + 0.
Werkt het blokschema nu goed?
klas
Nee. Het blokschema wil direkt al twee getallen lezen.
leerling
(Maakt met behulp van vraagblok na rekord := LEES ook hiervoor een voorziening).
docent
Nu staat er op getalkaarten:
+ 10.41 + 10.52 + 10.35 + 10.40 ..... + 0
en dat zijn de tijden gemaakt in de wedstrijd hardlopen over 100 m. Wat is nu het rekord?
klas
Het kleinste getal.
docent
Wat moet gewijzigd worden in het blokschema om zo'n kleinste getal te vinden?
klas
(na enig zwijgen). Alleen het groter-teken moet een kleiner-teken worden.
docent
Zijn er nog opmerkingen?
leerling
Wat gebeurt er als zo'n grootste getal (of kleinste getal) twee keer voorkomt?
docent
Dan is het rekord blijkbaar geëvenaard, maar dat verandert niets aan het rekord. Probeer het blokschema maar eens met de getallen + 76 + 80 + 81 + 77 + 81 + 0.
leerling
Je weet nu wel wat het rekord is, maar niet wie het behaald heeft.
docent
Dat is waar. We zullen de volgende keer een opgave maken waarbij we de rugnummers zullen betrekken, zodat we ook het rugnummer van de rekordhouder weten. (Compk opgave 9.22). Verdelen jullie je nu in groepjes van twee. Eén schrijft een programma voor het vinden van het grootste, de ander voor het vinden van de kleinste van een aantal getallen. Schrijven jullie voor elkaar de getalkaarten en onthoudt welk antwoord er volgende week bij de ander uit moet komen.

Men lette er bij Compk opgave 9.22 op dat een programma volgens:



een onjuiste oplossing biedt voor het geval het eerste getal het grootste getal is.

Het programma van Compk opgave 9.23 mag niet beginnen met

```
START
grootste := LEES
kleinste := LEES
```

gezien bijvoorbeeld de getallenrij

+ 1 + 3 + 2 + 2 + 2 + 0.

2.1.12 De rij

(Compk hfdst 10, Wrks hfdst 11)

De rij wordt door veel leerlingen een moeilijk onderwerp gevonden. Dit heeft ondermeer de volgende oorzaken:

- het nut van de rij is niet duidelijk
- er zijn bijna geen eenvoudige opgaven, waarin de rij onmisbaar is.

Het nut van de rij wordt duidelijk gedemonstreerd aan de volgende opgave (ongeveer Compk opg. 10.1)

Op getalkaarten staan 100 getallen. Laat een lijst schrijven waarin staat aangegeven hoever elk getal van het gemiddelde van die 100 getallen afligt (onder hoever verstaan we nu maar even getal min gemiddelde zonder de absolute waarde te nemen).

- Elk van de honderd getallen toekennen aan variabelen a,b,c,d enz. enz. is mogelijk. Dit zou alleen nut hebben als de variabelen verschillende bewerkingen zouden ondergaan. Het is omslachtig om aan te geven: doe met variabele c hetzelfde als met a.
- Elk van de honderd getallen toekennen aan dezelfde variabele is mogelijk voor het berekenen van het gemiddelde.

```
      START
      t := 0
      som := 0
1     t := t+1
      a := LEES
      som := som+a
      ALS t<100 DAN 1
      gem := som/100
      KLAAR
```

Na deze berekening is de getallenrij niet meer beschikbaar. De opgave kan zo niet worden opgelost.

- De rij is vrijwel noodzakelijk als men een aantal variabelen eenzelfde behandeling wil laten ondergaan, waarna de waarden nog steeds beschikbaar moeten zijn.

```
      START
      RIJ(1:100)a
      t := 0
      som := 0
1     t := t+1
      a(t) := LEES
      som := som+a(t)
      ALS t<100 DAN 1
      gem := som/100
      t := 0
2     t := t+1
      SCHRIJF := a(t)-gem
      ALS t<100 DAN 2
      KLAAR
```

Eenvoudige opgaven waarin een rij gebruikt moet worden zijn helaas zeldzaam. Opgaven zoals § 2.3 nummer 45 zijn als oefening bruikbaar, maar weinig overtuigend. De nummers 42 en 44 zijn moeilijker maar daarentegen schoolvoorbeelden van nuttig rijgebruik.
Het komt voor dat leerlingen, nadat de rij behandeld is, te onpas gebruik van rijen maken. Bijvoorbeeld elke verzameling op getallenkaarten wordt, ongeacht de bewerking die zij moet ondergaan, opgeslagen in een rij. Hiertegen zou een waarschuwend geluid op zijn plaats zijn.
Het verdient aanbeveling om bij minder begaafde leerlingen de rij vrijwel alleen ten tonele te voeren in verband met bestanden.

2.1.13 Bestanden

(Compk § 13.3, Wrks hfdst 11)

Het achtergrondgeheugen biedt de mogelijkheid om gegevens op te slaan voor langer dan alleen de tijd dat het programma werkt. Het zelf opslaan van gegevens om daarover na één of meer weken weer te kunnen beschikken, is voor leerlingen niet van groot belang, hoewel niet onmogelijk. Het grote voordeel in de les is dat de leerling zijn programma's kan laten opereren op gegevens die hem werkelijk onbekend zijn maar onmiddellijk ter beschikking staan. De programma's winnen aan realisme voor de leerling wanneer hij niet zelf of zijn buurman de gegevens in de vorm van getalkaarten aan het programma hoeft toe te voegen.
Een probleem voor veel leerlingen is het uit elkaar houden van de opdrachten rijnaam := BESTAND(n) en BESTAND(n) := rijnaam.
Als docent kunt U in Uw voorbeelden deze opdrachten bijvoorbeeld uitspreken als: (de rij) rijnaam := (gevuld met de getallen uit) BESTAND(n) en BESTAND(n) := (gevuld met de getallen uit de rij) rijnaam.

Bij onderwerpen als girodienst, vliegtuigreservering (Compk § 15) en personeelsadministratie komt het gebruik van bestanden op elementaire wijze

naar voren. Van het laatste onderwerp tenslotte nog een klassegesprek.
De docent zorgt dat de leerboekjes gesloten blijven.

docent
We gaan een administratie opzetten voor een bedrijf met 200 werknemers. Dat wil zeggen dat we van elke werknemer bepaalde gegevens gaan onthouden. Natuurlijk willen we van elke werknemer weten wat zijn naam en adres is. Die staan op een lijst, genummerd van 1 tot 200. Wat moet er nog meer bekend zijn op het bedrijf van elk van deze mensen?

leerling
De salarissen.

docent
Is dat nodig?

leerling
Er moet toch bekend zijn wat er aan elk moet worden uitbetaald aan het eind van de maand.

docent
Juist. Wat nog meer?

leerling
De leeftijden.

docent
Is dat nodig?

leerling
Je moet weten wanneer iemand met pensioen gaat.

docent
Goed. Wat nog meer?

leerling
Of het een man of een vrouw is.

docent
Ik kan me voorstellen, dat je dat zelf graag wilt weten, maar is het ook nodig voor onze administratie?

leerling
Misschien niet.

docent
Iets anders dan.

leerling
Of hij getrouwd is, ja of nee.

docent
Waarom dat dan?

leerling
Dat scheelt in de belasting.

docent
En wat heeft onze administratie met die belasting te maken?

leerling
Die wordt toch van het salaris afgetrokken?

docent
Juist, de loonbelasting wordt op het salaris ingehouden.

leerling
Maar een vrouw betaalt toch minder belasting dan een man.

docent
Een werkende gehuwde vrouw betaalt inderdaad minder, wat wil je met deze opmerking?

leerling
Dan moet de administratie ook weten of het een man of een vrouw is.

docent
Goed, we noteren dus ook het geslacht. Even verzamelen: behalve naam en adres administreren we salaris, leeftijd, geslacht, al of niet gehuwd zijn. Iemand nog meer?

leerling
Hoelang hij bij de firma werkt.

docent
Is dat nodig?

leerling
Voor jubilea.

docent
Goed, dat er dus ook bij. Je hebt wel gemerkt dat ik telkens vroeg: "is dat nodig?". Als je een administratie gaat opbouwen, moet je je eerst afvragen welke gegevens je nodig hebt. Daarna hoe we deze gegevens zullen opbergen. De 200 salarissen brengen we onder in bestand 19. Laten we aannemen dat ze afgerond zijn op gehele guldens, bijvoorbeeld +1071+1280+1368+.....enz. Waarom is het niet erg handig om op dezelfde manier de leeftijden op te bergen?

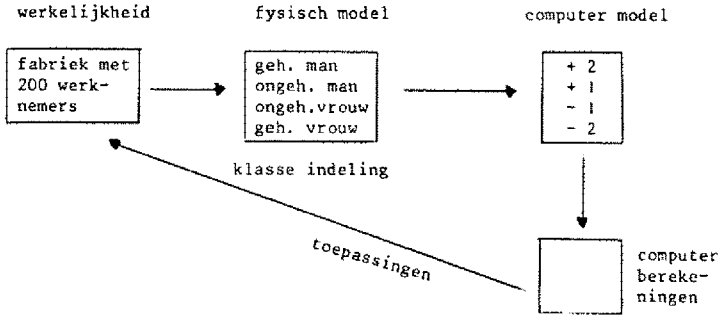
leerling
Als we de geboortejaren noteren, dan hoeven we niet meer te wijzigen. De leeftijden veranderen steeds.

docent
Goed zo. Je ziet, ook de manier van opbergen van gegevens kan meer of minder handig. Nu het geslacht en het gehuwd zijn. Dit zijn gegevens die niets met getallen te maken hebben. Toch kunnen we in ons bestand alleen getallen opnemen. Hoe lossen we dat op

---

We verlaten hier de klas. De rest van het gesprek laat zich raden. Men lette op de laatste alinea waarin een representatieprobleem ter sprake komt. Niet numerieke gegevens moeten door getallen worden gerepresenteerd. Bij de behandeling van "grote" computertoepassingen komt het maken van een model van de werkelijkheid duidelijk naar voren. Men kan geen berekening maken aan de werkelijkheid zelf, maar de werkelijkheid moet gerepresenteerd worden door een computermodel en aan dit model worden de berekeningen gemaakt. Naast een voorbereiding op de maatschappij waarin computertoepas-

singen niet meer zijn weg te denken en naast het leren relativeren van de uit deze toepassingen verkregen resultaten (deze zijn niet beter dan het model waaraan de berekeningen plaats vonden) is dit proces een eenvoudig en zeer illustratief voorbeeld van wat men in de wiskunde onder "matimatiseren" verstaat.




```
START
a := LEESSYM
b := 10
SCHRIJFSYM := a
NR
TEKST := "a"
NR
SCHRIJF := a
NR
SCHRIJFSYM := b
NR
TEKST := "10"
NR
SCHRIJFSYM := b+30
NR
TEKST := "b+30"
NR
SCHRIJFSYM := a+b
NR
SCHRIJF := a-b
NR
TEKST := "a-b"
KLAAR
b
```

In Compk opgave 14.1 en enkele volgende is niet exact te bepalen welke leeftijd de werknemers hebben uit het bestand met de geboortejaren. Dat hangt immers van de datum af, waarop de peiling plaats vindt. Laat U Uw leerlingen dit zelf ontdekken. Alleen een feest op oudejaarsavond (Wrks 12-7) geeft zekerheid omtrent de leeftijden.

2.1.14 Manipuleren met letters (Compk §16)

Een gewild onderwerp bij veel leerlingen, omdat het speelse elementen combineert met een groot aantal toepassingen.
Op twee toepassingen wordt iets nader ingegaan.
Bij de eerste behandeling van dit onderwerp ontstaan bij leerlingen nog weleens misverstanden over de interne representatie, bijvoorbeeld de volgende.

- De symbolen staan zelf in de geheugenplaatsen.
- De computer herkent de getalrepresentatie als afkomstig van een letter, je mag er vast niet zomaar mee rekenen.

Oefeningen als de volgende kunnen dit snel oplossen. Wat wordt achtereenvolgens afgedrukt door het volgende programma:

De invoering van de schrapkaart heeft figuur 16.2 minder van toepassing gemaakt. Vraagstuk 16.4 levert ook daardoor problemen. Het symbool ≥ aangestreept op de schrapkaart wordt vertaald in twee tekens >=, met achtereenvolgens de interne waarden 86 en 84. Symboolwaarde 88 kan dus (nog) niet via een symbool op een schrapkaart worden ingevoerd. SCHRIJFSYM := 88 heeft echter wel degelijk het afdrukken van ≥ ten gevolge.

Naar aanleiding van opgave 16.11 kan ter sprake komen dat het zetten van kolommen veelal door computers wordt geleid. Opgave 16.12 gaat hierop dieper in. In werkelijkheid zal men op het eind van de regel ook het afbreken van woorden toelaten. Men kent danook programma's die zonodig nederlandse woorden kunnen afbreken. Het voert te ver om dergelijke programma's te behandelen. Wel kan gewezen worden op de beperkingen van deze programma's. Het is al heel moeilijk om een computer te laten bepalen hoe "kwartslagen" moet worden afgebroken: het zal van de context afhangen of "kwart-slagen" of "kwarts-lagen" bedoeld werd. Leerlingen kunnen gemakkelijk in kranten de fouten opsporen ten gevolge van het automatisch kolommen zetten.
Voorbeeld:

vleermuizen en vogels. Onder de reptielen zijn er een paar die ruim honderd miljoen jaar geleden vleugels hebben gekregen. Bij die vliegende reptielen is slechts de ringvinger sterk verlengd en de vlieghuid is slechts daaraan bevestigd. Duim, wijs- en middelvinger blijken kleine stompjes met sterkgekromde nagels te zijn. De pink is bij het

Bij het volgende is duidelijk te zien dat het programma er vanuit gaat dat "on" altijd een lettergreep is.

Iets dergelijks treft men bij de grote uitgestorven reptielen aan: geweldig ontwikkelde achterpoten, maar voorpootjes die alleen nog maar kleine, onbetekenende aanhangsels zijn. Dit

Naar aanleiding van §16 kan ter sprake komen het gebruik van computers in de taalwetenschappen. Hierover het volgende.

Onder klassifikatieprocessen verstaan wij het omwerken van een tekst in *een gegeven orde* (lopende tekst, woordenboek of frekwentielijst bv.) *tot een andere orde*. Dit is voor de computer de meest elementaire bewerking.

Indexen en Concordanties

Een *index* is de (meestal) alfabetische ordening van de woorden van een tekst voorzien van de plaats(en) die ze in die tekst hebben. Een *concordantie* verschaft niet enkel vindplaatsen maar geeft ook een hoeveelheid context. Deze varieert naar gelang van het programma dat de computer instrueert. In de *Woordindex op Mei van Gorter* vinden wij de woorden alfabetisch geordend met opgave van de bladzijde, de versregel en het nummer van het woord in de regel. Zo vinden wij *koel* aldus opgegeven:

Koel (frekw. = 8) 19-16-5; 33-30-6; 82-26-6;
101-22-1; 124-06-1; 155-07-2;
155-07-7; 156-09-9.

Bij een concordantie waarbij de lengte van de context zou begrensd worden door de twee meest nabije zintekens waartussen het woord staat, zouden we verkrijgen:

koel 19-16-5, en laat zijn nog zijn voeten koel in dauw van wei.
33-30-6 - koel lag nog de wel -
82-26-6 - een hinde gelijkend draafde ze uit een koel woud.
101-22-1 ? Koel is de maan,
enz.

Door dit voorbeeld zal het nut van indexen en concordanties wellicht duidelijk(er) geworden zijn; onderzoek naar de combinatie- en associatiemogelijkheden van een woord (zijn semantisch-stilistisch veld), naar temata, motieven en beeldspraak bij een auteur, syntaktisch onderzoek (zoals het gebruik van pronomiale verwijzingen of de combinatiemogelijkheden van structurele woorden) e.d. kunnen nu meer dan ooit op een betrouwbare, want exhaustieve wijze onderzocht worden.

Frekwentielijsten

De computer kan van om het even welke tekst niet alleen een index of een concordantie maar ook een frekwentielijst maken. Het volstaat dat hij de woorden alfabetisch rangschikt en identieke woorden samentelt.

Dit kan dan het uitgangspunt worden voor verder onderzoek. Bij het vervaardigen van de frekwentielijst op *Mei* ontdekten wij bv. dat het woord *koel* (+ *koele*, *koelen* (a.), *koeler*) 17 maal voorkwam. Daarnaast kwamen ook nog verwante woorden voor als *koelende* (1x), *koelte* (4x), *koelvat* (1x). Een woord als *fris(ser)* daarentegen kwam slechts éénmaal voor.

Het zou interessant zijn deze frekwentie te kunnen vergelijken met de frekwentie die *koel* en *fris* in een algemene frekwentielijst voor het Nederlands zouden bezitten.

Een grote algemene frekwentietelling van Nederlandse woorden en strukturen zou dus niet allen bij didaktisch, maar ook bij zuiver linguïstisch en literair onderzoek zeer waardevol zijn juist door het preciezer inzicht in normen en normafwijkingen dat hieruit zou kunnen gewonnen worden. Een dergelijke lijst blijft echter ondanks de mogelijke computerhulp nog steeds uit.

Retrograde lijsten

Bij een retrograde lijst worden de woorden alfabetisch gerangschikt naar de eindletter i.p.v. naar de beginletter zoals gebruikelijk. Men bekomt aldus een lijst die voor het Nederlands bv. in het begin worden zou bevatten als: *tuba*, *logica* en *mica* en op het einde *ersatz*, *hertz* en *jazz*. Verder zouden woorden eindigend op *-man*, *-ig*, *-heid*, *-tuig*, *-ing*, enz. allen samengegroepeerd zijn. Het nut van dergelijke lijsten ligt dus vooral op het gebied van de *woordvorming*. Door retrograde rangschikking immers komen woorden met hetzelfde eindmorfeem samen.

Vergelijking inzake suffixen tussen talen onderling (met eventueel onderzoek van interferentieverschijnselen zoals het gebruik in het Nederlands van *-atie* bij woorden waar het Frans geen *-ation* kent) is mogelijk, aangezien voor de meeste (althans Westerse) kultuurtalen reeds een dergelijk instrument bestaat. Verder kan een retrograde lijst ook dienst bewijzen als rijmwoordenboek en als hulp bij reconstructie van teksten.

Waar, zoals gezegd, voor de meeste talen *retrograde woordenboeken* bestaan, zijn *retrograde frekwentielijsten* nog vrij zeldzaam. Daarmee bedoelen wij de retrograde rangschikking van een algemene frekwentielijst. Dit schijnt ons nochtans een waardevolle aanvulling te zijn op het retrograde woordenboek. Aan de hand van een retrograde frekwentielijst kan men immers een probabilistisch *model* van taalgebruik voor sufsixen of andere eindmorfemen opstellen waartegenover individuele "performances" kunnen worden afgewogen. Zo konden wij konstateren dat de diminutiefvorming bij Gorter (Mei) afweek van het gewone Nederlands: waar in de retrograde (algemene) frekwentielijst geen verkleinwoorden op *-ke* voorkwamen troffen wij bij Gorter: *kindeke* (f=1), *maankindeke* (f=1), *windeke* (f=2), *bijke* (f=1), *woordekens* (f=1) en *zusterke* (f=1) aan.

2.1.15 Sorteren en zoeken

(Compk §17)

Een wat moeilijker onderwerp dat echter een indicatie geeft voor efficientieproblemen bij computertoepassingen. Bij de girodienst kwam in 13.5 al de noodzaak van sorteren naar voren.
Het dagelijks sorteren van rijen ter lengte van enkele miljoenen getallen stelt hoge eisen aan de werkingssnelheid van de algoritme. Het verdient misschien aanbeveling ook andere algoritmen voor sorteren te behandelen.
In 17.1 werd beschreven het sorteren door het naar de n-de plaats schuiven van het grootste van n getallen, n loopt van het oorspronkelijke aantal terug tot 2. Een andere sorteermethode verwisselt steeds (zo nodig) twee na elkaar komende getallen.

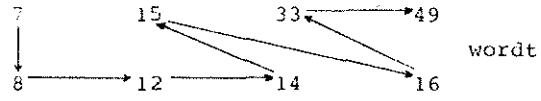
15	13	13	13	13	13	7	7
13	15	15	15	7	7	13	12
49	49	7	7	15	12	12	13
7	7	49	12	12	15	15	15
12	12	12	49	49	49	49	49

Deze methode is iets onvoordeliger, omdat na het vergelijken van twee getallen er een verwisseling (3 opdrachten) kan plaats vinden, terwijl bij het zoeken naar de grootste van een rij getallen na het vergelijken slechts een vervanging (1 opdracht) noodzakelijk kan zijn. Het programma voor de hier beschreven methode heeft een redelijk eenvoudig uiterlijk.

```
   .........
   .........
   MET -1, j := n, 2
   MET 1, i := 1, j-1 <------------ vlag := 0
   ALS a(i) ≤ a(i+1) DAN 26
   p := a(i)
   a(i) := a(i+1)
   a(i+1) := p
26 HERHAAL <------------------------ vlag := 1
   HERHAAL <------------------------ ALS vlag = 0 DAN 27
27 ........
```

Men kan bovendien een verfraaiing aanbrengen (de opdrachten met vlag) waardoor de algoritme stopt op het moment dat geen verwisselingen meer plaatsvinden (vlag blijft 0).
Er wordt geen overbodig werk verzet in een rij getallen die al op de juiste wijze gesorteerd blijkt te zijn. Ook hierbij ligt het aantal uitgevoerde opdrachten in de orde van $2\frac{1}{2}n^2$.

Een in de praktijk gebruikte algoritme is ingewikkelder. Men gaat hierbij uit van de gedachte dat het aantal opdrachten om twee gesorteerde rijen met lengte $\frac{1}{2}n$ in elkaar te schuiven tot een gesorteerde rij met lengte n van de orde n is.



7 8 12 14 15 16 33 49

Van elk getal kan men de opvolger bepalen door één opdracht, nl. het vergelijken van zijn oorspronkelijke opvolger en het kleinste getal uit de andere rij dat nog niet aan de beurt geweest is.
Het netjes in een rij zetten en het bijhouden van een kleine administratie vereist nog wel enkele opdrachten, maar gaat in het totaal toch niet 8 te boven.
Als men uitgaat van $n = 2^k$ te sorteren getallen, dan kan men deze rij opdelen in 2^{k-1} tweetallen en al deze tweetallen sorteren ten koste van $2^{k-1} \times 8 \times 2 = 8n$ opdrachten. Deze 2^{k-1} rijtjes kunnen twee aan twee in elkaar geschoven worden tot 2^{k-2} gesorteerde viertallen, ten koste van $2^{k-2} \times 8 \times 4 = 8n$ opdrachten weer twee aan twee in elkaar schuiven tot 2^{k-3} gesorteerde achttallen ten koste van $2^{k-3} \times 8 \times 2^3 = 8n$ opdrachten. Dit behoeft hoogstens k maal herhaald te worden om tot een gesorteerd n-tal te komen, d.w.z. een volledig gesorteerde rij.

In totaal verbruikt men 8nk opdrachten. Het op deze wijze sorteren van een rij ter lengte van een willekeurige n kost in het algemeen in de orde van $8n.\log n$ opdrachten, wat voor grote n aanzienlijk vlugger gaat dan een algoritme van de orde n^2. Vergelijk verder Compk §17.3.

2.1.16 S i m u l a t i e

(Compk §18.3, Wrks hfdst 14)

Een mijns inziens belangrijk onderwerp in het geheel van de computerkunde, omdat hierbij de modelvorming duidelijk naar voren komt. Als voorbeeld Wrks 14-8.

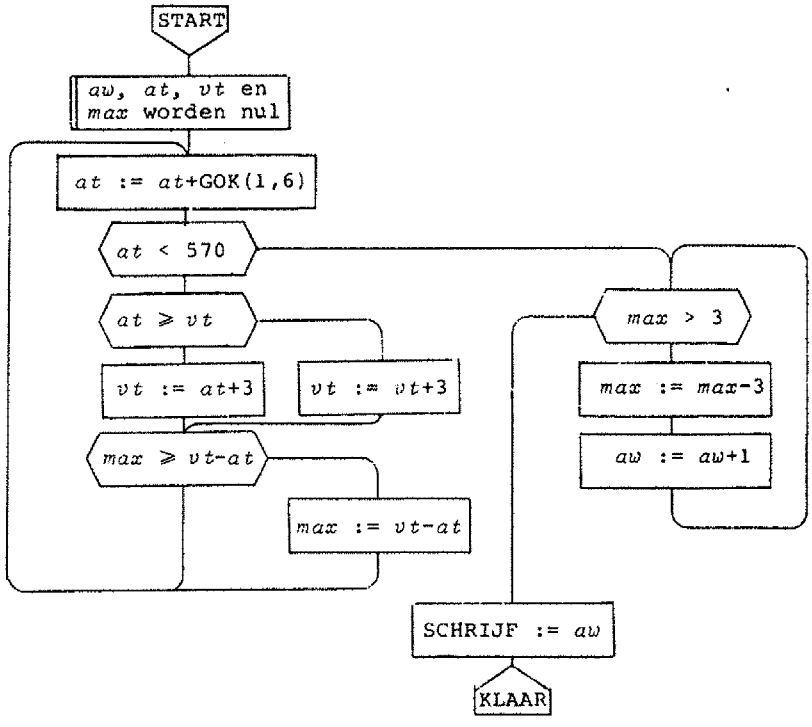
Iemand wil een uitkijktoren bouwen. Er komt een lift die elke minuut éénmaal op en neer gaat en waarin ten hoogste 20 mensen kunnen plaatsnemen. Simuleer de situatie door het aantal mensen in de lift bij elke tocht (naar boven zowel als naar beneden) te gokken, met de volgende uitzondering. De eerste drie minuten gaat de lift leeg naar beneden.
Zet de lift een uur en bedrijf en laat het grootste aantal mensen bovenin de toren bepalen. Hieruit zou de bouwer van de toren enigszins kunnen afleiden voor hoeveel mensen er ruimte moet zijn op de bovenste etage.
Denk eraan dat de lift nooit meer mensen naar beneden kan halen dan er boven aanwezig zijn.

```
    START
    max := GOK(0,20) + GOK(0,20) + GOK(0,20)
    aant := max
    MET 1, k := 4,60
    aant := aant + GOK(0,20)
    ALS aant > max DAN 1 ANDERS 2
1   max := aant
2   aant := aant - GOK(0,20)
    ALS aant < 0 DAN 3 ANDERS 4
3   aant := 0
4   HERHAAL
    TEKST := "het grootste aantal mensen in de
                                          toren is"
    SCHRIJF := max
    KLAAR
```

De noodzaak van simuleren volgt uit het feit dat de uitkijktoren nog niet aanwezig is en er dus geen mogelijkheid is om uit de werkelijkheid een konklusie te trekken. De beperkingen van het model worden in de opgave al gesuggereerd door 'enigszins kunnen afleiden' enz. Het is immers niet gezegd dat men bovenin de toren beslist ruimte moet hebben voor het maximum aantal te verwachten mensen. Hierbij zullen de financiën ook een rol spelen. Vrijwel elke leerling zal aanvoelen dat de simulatie niet voorziet in een heldere zonnige vakantiedag waarop de lift zolang mogelijk vol naar boven zal gaan.
De eerste drie keer gaat de lift leeg naar beneden omdat we aannemen dat iedereen tenminste 3 minuten van het uitzicht wil genieten alvorens weer in de lift te stappen. Maar het bovenstaande programma houdt gedurende de rest van het uur hiermee geen rekening. Het kan voorkomen dat halverwege het uur het bovenin de toren leeg is en dat de eerstvolgende liftlading die boven komt grotendeels onmiddelijk weer vertrekt. Ook dat is allerminst realistisch. Een betere simulatie zou zijn om van elk bovengekomen individu het genot van het uitzicht te gokken tussen bijvoorbeeld 3 en 15 minuten. Hierbij moet dan weer rekening worden gehouden met de beperking tot 20 personen van de neergaande lift. Het uitprogrammeren van dit soort verbeteringen lijkt me veel minder belangrijk dat een goede diskussie in de klas over de bezwaren tegen het model en de richting waarin verbeteringen mogelijk zijn.

De opgaven Compk 18.9 en 18.10 geven aanleiding tot wachttijd problemen. De betreffende programma's zijn beslist minder eenvoudig te noemen. Als voorbeeld van een uitgewerkt vraagstuk het volgende. (Cursus Computerkunde voor Mavo- en LBO-docenten, opgave 19).
Bij een automatische autowasinrichting gebruikt men een lijn die vier minuten voor het schoonmaken van een auto nodig heeft. Men weet uit ervaring dat de tijd tussen de aankomst van een klant en de aankomst van de volgende een toevalsgetal is tussen 1 en 6 minuten. Een rij van wel meer dan tien wachtende auto's komt nogal eens voor. Het bedrijf overweegt de installatie van nieuwe apparatuur waardoor de doorvoersnelheid verhoogd wordt tot maximaal één auto per drie minuten. Simuleer een werkdag van dit vernieuwde systeem en laat het grootste aantal wachtende auto's afdrukken. Het bedrijf opent om 8.30 en sluit om 18.00 uur.
De oplossing van dit vraagstuk vindt U op de volgende bladzijde in blokschema. Hierin is:

at = aankomsttijd van een auto
vt = vertrektijd van een auto, in het schema vanaf *vt* := .... .
Voordien is het de vertrektijd van de voorgaande auto.
max = de langst voorkomende tijd tussen aankomst en vertrek van eenzelfde auto
aw = aantal wachtende auto's.



De homogene verdeling van de tijden tussen twee klanten door GOK(1,6) doet onnatuurlijk aan (zie ook opmerking aan het eind van Compk 18.3), want het is allerminst zeker dat binnen elke periode van 6 minuten tenminste één klant arriveert. Beter is uit te gaan van de kans op binnenkomen per seconde en wel als volgt:

De kans op binnenkomen in de 1e sec. $= \frac{1}{60} \times \frac{1}{6}$

De kans op geen klant in de 1e sec. $= 1 - \frac{1}{360}$

De kans op geen klant in de 1e 2 sec. $= (1 - \frac{1}{360})^2$

De kans op geen klant in de eerste n seconden =

$$(1 - \frac{1}{360})^n = ((1 - \frac{1}{360})^{360})^{\frac{1}{360}n} \approx (e^{-1})^{\frac{1}{360}n} = e^{-\frac{1}{360}n}$$

De kans P op een tussentijd van n seconden neemt sterk af bij grotere waarden van n. Men kan dit simuleren met behulp van de inverse funktie van

$y = e^{-\frac{1}{360}x}$ nl. $x = -360 \ln y$. Bij een regelmatig verdeelde y is de tussenperiode x volgens onze eisen verdeeld.

Kortom: x := -360 LN(GOKC) geeft een betere gesimuleerde tussentijd (in seconden uitgedrukt).

2.2 TIJDSCHEMA

Voor het behandelen van de leerboekjes Compk. 1 en 2 is het volgende tijdschema redelijk gebleken. Het is gebaseerd op één wekelijks lesuur. Hierbij bedenke men dat niet alle vraagstukken uit dit boekje behandeld hoeven te worden. Daarom volgt onder het tijdschema een lijst van vraagstukken, die als facultatief beschouwd kunnen worden en door de meeste leerlingen kunnen worden overgeslagen.

Compk. onderwerp	aantal lessen	paragr. nummer	tijdstip cursusjaar
Inleiding, algoritme, blokschema computer	2	1-3	begin sept.
Geheugen en rekenopdrachten	3	5	eind sept.
Schrijven en lezen	2	6	half okt.
programmeren	3	7	eind okt.
De vertakking	3	8	eind hov.
De cyclus	4	9	begin jan.
De rij	3	10	begin.febr.
formules + giro	3	12,13	begin mrt.
personeelsadm.	3	14	apr.
keuze onderwerp	uit	15-20	restant

(vervolg)

Wrks hoofstuk	aantal lessen	tijdstip cursusjaar
7	1½	
8	2	eind nov.
9	4	begin jan.
10	2	eind jan.
11	3	eind febr.
12	3	eind mrt.
13	2	apr.
herhaling en keuzenonderwerp		restant

Facultatieve opgaven

Compk.	3.12	3.13	8.10	8.11	8.12
	8.13	9.3	9.6	9.10	9.25
	9.26	10.3			

Tijdschema werkschrift

Wrks hoofdstuk	aantal lessen	tijdstip cursusjaar
1	2	begin sept.
2	1½	
3	1½	eind sept.
4	1	
5	2	
6	1½	eind okt.