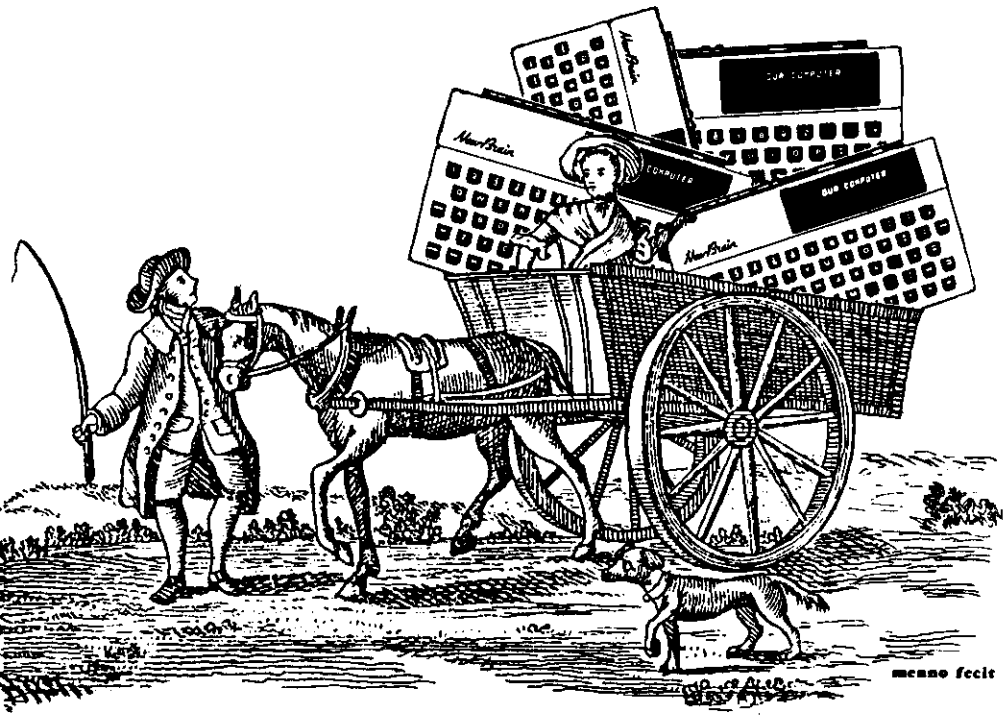
NewBraindag



11 april 1992
10 oktober 1992

sbbo leiden

NewBrain on-line

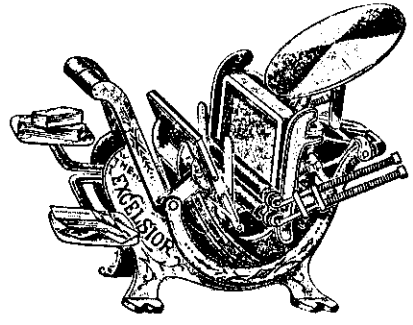
uitgave van de
NewBrain-
gebruikersgroep

17

april 1992

newbrain on-line is een uitgave van de nederlandse newbrain-gebruikersgroep. de bladen zijn licht gelijmd, zodat ze eenvoudig los te maken en in een ringband op te bergen zijn. de artikelen kunnen dan bij voorbeeld op onderwerp geordend worden. de perforatie is afgestemd op de ringband met hcc-opdruk, die bij de hcc te koop is.

abonnementen kunnen alleen beëindigd worden door schriftelijke opzegging bij de gebruikersgroep.
eerder verschenen nummers kunt u bestellen door overmaking van ƒ 10,- per nummer op girorekening 2505800 ten name van hcc newbrain-gebruikersgroep te amsterdam



kopij

wanneer er voldoende kopij ingestuurd is, zal het volgende nummer van newbrain on-line worden uitgebracht. stuur daarom uw kopij zo snel mogelijk in naar: newbrain-gebruikersgroep, postbus 4494, 1009 AL amsterdam

het is voor de redactie verreweg het makkelijkst de kopij aangeleverd te krijgen op cassette of diskette. dat bespaart de fouten bij het overtypen. vermeld bij een diskette wel het formaat (alle newbrainformaten en 360k ms-dos kunnen we aan). de tekst mag op alle gangbare manieren zijn opgeslagen: wordperfect-, texy-, wordstar-, wp- en alle mogelijke ascii-bestanden zijn bruikbaar. de cassette of diskette krijgt u natuurlijk terug. zeker als in een artikel een programma-listing van enige omvang voorkomt, is een cassette of diskette onontbeerlijk.
dit alles mag u natuurlijk niet beletten om kopij in te sturen. de redactie ontvangt liever kopij, waar ze wat meer werk mee heeft, dan helemaal geen kopij

Memo facit

New Brain on-line

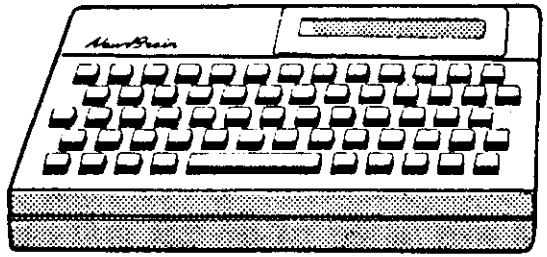
TEN GELEIDE

er is weer een newbrain on-line uitgekomen. maarten floor demonstreert een nieuwe toepassing van de buitenwereldkaart, en bas boetekees heeft een diepgaand onderzoek gedaan naar sorteren. het resultaat is een artikel, dat niet alleen voor newbrainers interessant is, maar voor iedereen die programmeert

vergeet niet, dat on-line bestaat bij de gratie van leden die kopij insturen!

de redactie

het bestellen van software
neem het bestelformulier achterin deze on-line, en vul daarop in, welke software u wenst te ontvangen (niet alleen de opgesomde delen zijn leverbaar, maar ook alle cp/m-volumes van de 'programmatheek' van de dos-gebruikersgroep!). kruis het gewenste formaat aan: cassette of diskette van 200k, 400k ss, 400k ds of 800k voor de newbrain, of vul het proton-formaat in. stuur de bon samen met een girobetaalkaart of betaal- of eurocheque (vergeet niet het nummer van giro- of betaalpas in te vullen) op naar postbus 4494, 1009 AL amsterdam. het verschuldigde bedrag kan ook worden voldaan door overschrijving op girorekening 2505800 ten name van hcc newbrain-gebruikersgroep te amsterdam. ook in dat geval het bestelformulier opsturen. zonder bestelformulier gaat het vast mis



NewBrain-
gebruikersgroep
postbus 4494
1009 AL amsterdam

voorzitter: jan wubben, (010) 4557698
secretaris-penningmeester: menno stevens, (020) 6924137
bibliothecaris: wim van hoek, (020) 6948430
bas boetekees, (03465) 76211
maarten floor, (02963) 4374
co koning, (02521) 12742
guus von morgen, (02158) 23381

postgiro 2505800 tnv hcc newbrain-gebruikersgroep, amsterdam

de newbrain-gebruikersgroep is een onderdeel van de
HCC **HOBBY COMPUTER CLUB**
inschrijvingsnummer kvk leiden: V445230

sorteren

In deze On-line wil ik eens dieper ingaan op het sorteren van databestanden in de taal Pascal. Voor het sorteren van data-arrays zijn er diverse methoden, zoals bubbel of normaal, quick, heap, recursief etcetera, maar voor databestanden op disk vind je geen enkele methode. Wanneer het geheugen toereikend is om het gehele databestand te laden, kunnen we onder andere methode 6 of 7 gebruiken, die gebaseerd zijn op tevoren gedeclareerde pointers of arrays. Voor onze NewBrain, die een beperkt geheugen heeft, zijn deze twee methoden niet echt toepasbaar voor databestanden die meer dan 4 à 5 Kbyte ruimte in beslag nemen. We moeten dan over disk gaan sorteren en wel met een methode die redelijk snel werkt.
Het doel van dit artikel is die methode te vinden.

NORMAAL SORTEREN (METHODE 1 EN 2)

```
For I := 1    to Max-1 Do
For J := I+1 to Max    Do
Begin
 If SortBuf(J) < SortBuf(I)
 then Wisselrecords(I,J)
End
```

figuur 1. normaal sorteren

Dit is in principe de simpelste vorm voor het sorteren van gegevens, maar het heeft een groot nadeel. Om de file te sorteren moeten alle data uit de file vaak gelezen worden, en dan vergeten we het aantal te wisselen records om op- of aflopend (ascending of descendig te sorteren.

Onze winst moeten we dus zien te halen uit het beperken van het aantal te lezen records en daarvoor kunnen we in deze methode een record in het geheugen van de computer bewaren.

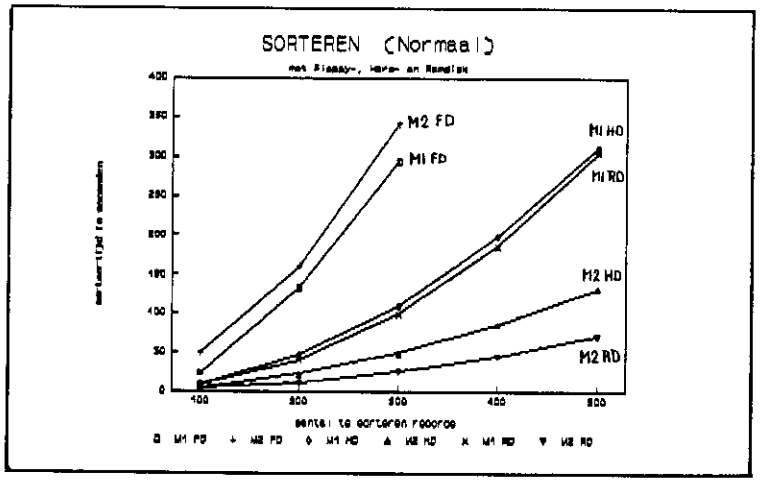
In de sorteerroutine van figuur 2 gebruiken we de variabelen IRecord en JRecord om de inhoud van beide records tijdelijk in het geheugen te bewaren.

In de I-loop wordt het I-de record gelezen en in IRecord opgeslagen. In de J-loop wordt het J-de record gelezen en in JRecord opgeslagen. Wanneer de tekst in JRecord kleiner is dan de tekst in IRecord, dan moeten (bij oplopend sorteren) de twee records van plaats verwisseld worden.

```
For I := 1 to Max-1 Do
Begin
 LeesRecord(I,IRecord);
 Gewisseld := False;
 For J := I+1 to Max Do
 Begin
  LeesRecord(J,JRecord);
  If JRecord.Tekst < IRecord.Tekst
  then
   Begin
    SchrijfRecord(J,IRecord);
    IRecord:=JRecord;
    Gewisseld := True
   End
 End
 If Gewisseld
 then SchrijfRecord(I,IRecord);
End
```

figuur 2. normaal sorteren (met geheugen)

Dit gebeurt door achtereenvolgens de inhoud van IRecord op disk, in het databestand, weg te schrijven; de inhoud van record J in het geheugen naar IRecord te kopiëren; en de boolese variabele Gewisseld te zetten. Wanneer de gehele J-loop doorlopen is, wordt er gekeken of er tijdens het sorteren gewisseld is. Is dit het geval, dan wordt IRecord vanuit het geheugen teruggeschreven in het databestand.



afbeelding 3. normaal sorteren

U zult zich misschien afvragen, wat het stukje programma in figuur 2 voor effect heeft ten opzichte van het programma in figuur 1. Deze vraag wil ik beantwoorden aan de hand van afbeelding 1. U ziet de gemeten tijden voor het sorteren van een aflopend naar een oplopend bestand. Het programma in figuur 1 gebruikt aanzienlijk meer tijd voor het sorteren dan programma 2, wanneer we gebruik maken van een ram- of harddisk. Bij gebruikmaking van een floppy disk is het effect geheel verdwenen, ja zelfs negatief. Het bewaren van records heeft dus alleen effect, als het gaat om het sorteren van databestanden op ram- of harddisk, en niet bij floppy disks.

Voor alle volgende sorteerroutines zullen we dit steeds moeten onderzoeken en dat was voor mij de reden dit artikel in On-line te schrijven. Als ik dacht een snellere sorteermethode gevonden te hebben, bleek dit in de praktijk soms knap tegen te vallen.

Let op: dit zijn tijden gemeten op een IBM-machine. Voor de NewBrain zullen de tijden minimaal met een factor 10 vermenigvuldigd moeten worden, omdat deze machine de sourcecode aanzienlijk langzamer verwerkt (het zou te veel tijd vergen deze tests op NewBrain of Proton-board uit te voeren om de gemiddelde tijd van 7 metingen te bepalen en handmatig te klokken).

SHELLSORTEREN (METHODE 3 EN 4)

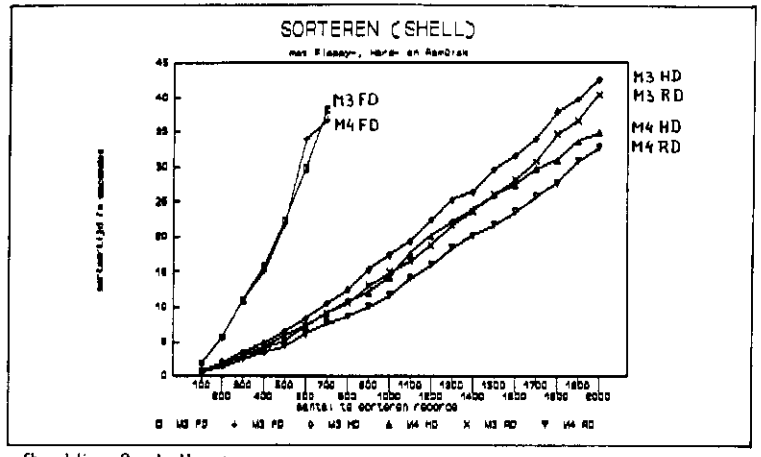
Shellsorteren heeft als principe, dat er op een bepaalde afstand records met elkaar vergeleken worden. Deze afstand krijgt als beginwaarde de helft van het aantal te sorteren records.

```
Afstand := Max Div 2;
While Afstand > 0
 For I := Afstand+1 To Max Do
 Begin
  J := I-Afstand;
  While J>0
  Begin
   L := J + Afstand;
   If Sortbuf(J) <= Sortbuf(L)
   then J:=0
   else Wisselrecords(J,L);
   J := J - Afstand;
  End;
 End;
 Afstand := Afstand Div 2;
End;
```

figuur 3. shellsorteren

Stel dat we 10 records in een databestand hebben, dan is de afstand 10 Div 2 = 5 en vergelijken we record 1 met 6, 2 met 7, 3 met 8, 4 met 9 en 5 met 10. Wanneer deze run is uitgevoerd staan de kleine records (gevuld met tekst) in de sorteervolgorde bovenaan. We verkleinen nu de afstand tot (5 Div 2 =) 2 en herhalen de vergelijking tussen record 1 met 3, 2 met 4, tot 8 met 10. We herhalen dit, totdat de afstand 0 is.

Met deze routine is het slechts mogelijk records in het geheugen te bewaren, als de afstandvariabele 1 is, omdat dan record X met X + 1 wordt vergeleken en in de volgende stap X + 1 met X + 2. We zouden dus om de leesbe-



afbeelding 2. shellsorteren

weging over de disk te beperken de inhoud van X + 1 naar die van X kunnen kopiëren. De tijdwinst zal echter niet spectaculair groot zijn, mede doordat beide records in hetzelfde diskbuffergeheugen van de computer staan.
De meetresultaten in afbeelding 2 bevestigen ons vermoeden. De sorteertijden van methode 3 en 4 zijn nagenoeg gelijk. Het lijkt verstandig de inhoud van de records J en L niet te bewaren en de vergelijking die steeds moet controleren of de afstand 1 is, geheel te verwijderen uit de sorteerprocedure van methode 3.

Vergelijken we deze methode (shell) met het normaal of bubbelsorteren, dan blijkt deze methode duidelijk sneller.

LABELSORTEREN (METHODE 5)

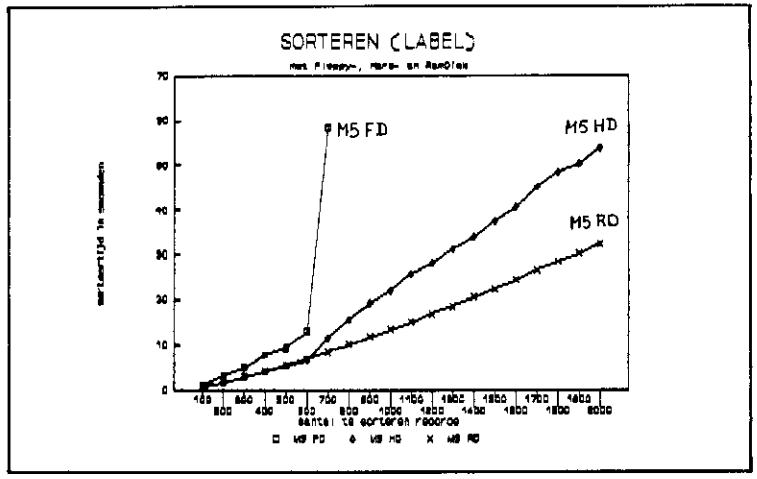
Deze sorteermethode vond ik in een Engelstalig boek 'Turbo Pascal Programmers Toolkit'. In het boek wordt geclaimd, dat deze sorteermethode even snel is als recursief sorteren. Ik dacht, dit kan ik de NewBrain-gebruikers niet onthouden.
Menig programmeur zal denken: een routine met labels, dat is noch Basic noch Pascal. Toegegeven, maar de auteur van het boek stelde dat de routine perfect werkt en ook snel is, dus maak ik een uitzondering. Een bijkomend voordeel is dat Basicprogrammeurs deze sorteerroutine ook kunnen gebruiken door de labels te vervangen door regelnummers; en dat is toch ook een groep die we sorteerroutines niet kunnen onthouden, omdat vooral zij gebaat zijn

Label	Source
H1	`L := Max Div 2 + 1`
	`R := Max`
H2	`If L > 1`
	`then Begin`
	`  L := L -1;`
	`  LeesRecord(L,DitRecord)`
	`  End`
	`else Begin`
	`  LeesRecord  (R,DitRecord)`
	`  LeesRecord  (1,WRecord  )`
	`  SchrijfRecord(R,WRecord  )`
	`  R := R - 1`
	`  If R = 1  { Bestand gesorteerd }`
	`  then Begin`
	`  SchrijfRecord(1,DitRecord)`
	`  Goto H0;  { of Exit Turbo 4 ev }`
	`  End`
	`  End`
H3	`J := L`
H4	`I := J`
	`J := J + J`
	`If J > R then goto H8`
	`If J = R then goto H7`
H5	`If SortBuf(J) < SortBuf(J+1)`
	`then J := J + 1`
H7	`LeesRecord  (J,WRecord)`
	`SchrijfRecord(I,WRecord);`
	`Goto H4`
H8	`J := I`
	`I := J Div 2`
H9	`If (DitRecord <= SortBuf(I)) Or`
	`  (J = L)`
	`then Begin`
	`  SchrijfRecord(J,DitRecord)`
	`  Goto H2`
	`  End`
	`else Begin`
	`  LeesRecord  (I,WRecord)`
	`  SchrijfRecord(J,WRecord)`
	`  Goto H8`
	`  End`
H0	`Data bestand is nu gesorteerd`

figuur 3. labelsorteren

bij snelle sorteerroutines, aangezien Basic aanzienlijk langzamer is dan Pascal.

Wanneer we de sorteertijden in afbeelding 3 bekijken lijken de sorteertijden



afbeelding 3. labelsorteren

redelijk. We ontdekten hier onze eerste afwijking tussen de verschillende sorteermethoden bij verschillende aantallen te sorteren records. Tot ongeveer 600 records werkt de labelmethode sneller dan de shellmethode en daarna gaat er duidelijk iets mis. De oorzaak is op dit moment niet duidelijk.

U zult misschien zeggen een meetfout? Nee, ik heb dit verschillende keren gemeten: de resultaten bleven gelijk en het aantal lees- en schrijfbewegingen om het bestand te sorteren zijn bij alle disktypes gelijk. Wanneer we naar de harddisktijden kijken, treedt ook daar hetzelfde effect op bij 600 records, maar bij de ramdisk zit de knik bij 2200 records. Ik zal hier later op terug-komen, wanneer de oorzaak hiervan bekend is. Op dit moment moet de conclusie luiden, dat deze sorteerroutine niet voldoet aan de verwachtingen.

RECURSIEF SORTEREN (METHODE 8)

Met recursief sorteren wordt een methode aangeduid, waarbij de procedure vanuit de procedure zelf aangeroepen wordt. Hierbij worden de plaats van de aanroep en de procedurevariabelen op de stack van de computer bewaard, zodat iedere keer na het verlaten van de procedure de laatste plaats van de aanroep kan worden teruggegeven en de procedure hier zijn werk weer voortzet (we kunnen dit vergelijken met de GOSUB-opdracht in Basic). Wanneer we deze procedure voor diskbestanden gebruiken, zal er bij enige honderden

te sorteren records een 'stack overflow' optreden. Dit heeft te maken met de hoeveelheid gegevens die de computer moet opslaan (PUSH'en) om ze later weer terug te halen (POPpen). We zien dat de recursieve aanroep twee maal wordt herhaald bij recursieve aanroep #1 en #2 of #3 en #4.

```
Sorteer(Laag,Hoog);          { Aanroep Procedure }

I:=Laag
J:=Hoog
LeesRecord(J,DitRecord);
Repeat
 While (I<J) And (SortBuf(I) <= DitRecord
 Do I:=I+1;
 While (J>I) And (SortBuf(J) >= DitRecord
 Do J:=J-1
Until I>=J
WisselRecord(I,Hoog)
If (I-Laag) < (Hoog-I)
then Begin
       Sorteer(Laag,I-1)   { Recursieve aanroep #1 }
       Sorteer(I+1,Hoog)   { Recursieve aanroep #2 }
     End
else Begin
       Sorteer(I+1,Hoog); { Recursieve aanroep #3 }
       Sorteer(Laag,I-1); { Recursieve aanroep #4 }
     End

Einde van de procedure                          #5
```

figuur 5. recursief sorteren

Bij het verlaten van de procedure zal er dus gekeken moeten worden, of alle op de stack gezette aanroepen er ook weer afgehaald zijn door de computer. Is dat niet het geval, dan moet hij op de plaats, die de stack aanwijst, de procedure vervolgen en uiteraard deze gegevens uit de stack verwijderen.

Dit laatste bracht me op het idee om de gegevens in een pointervariabele op te slaan en bij de recursieve aanroep labels te plaatsen, zodat we weten, waar we de procedure naar terug moeten sturen om de sorteercyclus af te maken.

```
Type Type_Ptr =    Record
                     Laag: Integer; { aanroepwaarden }
                     Hoog: Integer;
                     I: Integer; { tellerwaarde }
                     Punt: Byte; { plaats van aanroep }
                   End
```

```
Label   Sorteer(Laag,Hoog);          { Aanroep Procedure }
        H1,H2,H3,H4
-------------------------------------------------------------
H0;     If Laag < Hoog then Goto H5
        I:=Laag
        J:=Hoog
        LeesRecord(J,DitRecord);
        Repeat
         While (I<J) And (SortBuf(I) <= DitRecord
         Do I:=I+1;
         While (J>I) And (SortBuf(J) >= DitRecord
         Do J:=J-1
        Until I>=J
        WisselRecords(I,Hoog)
        If (I-Laag) < (Hoog-I)
        then Begin
              PushWaarden(Laag,Hoog,I,1)
              Hoog:=I-1
              Goto H0              { Recursieve aanroep #1 }
H1:           Laag:=Ptr^.I+1
              Hoog:=Ptr^.Hoog
              Goto H0              { Recursieve aanroep #2 }
             End
        else Begin
              PushWaarden(Laag,Hoog,I,3)
              Laag:=I+1
              Goto H0              { Recursieve aanroep #3 }
H3:           Laag:=Ptr^.Laag
              Hoog:=Ptr^.I-1
              Goto H0              { Recursieve aanroep #4 }
             End
-------------------------------------------------------------
H5:      If Ptr<>Nil
         then Begin
               Case Ptr^.Punt Of
                1: Goto H1;
                3: Goto H3;
               End;
               PopWaarden(Laag,Hoog,I);
               Goto H5;
              End;
        End;
```

figuur 6. recursief sorteren met labels

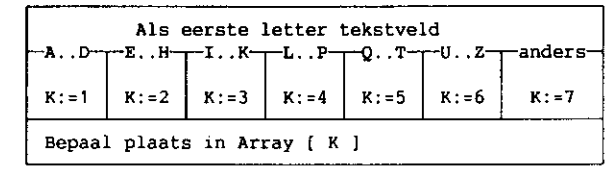
We kunnen nu zonder probleem 2000 records of meer sorteren. Wel dienen we nu zelf bij te houden, of er genoeg geheugenruimte is op de heap. Een alternatief kan zijn de adressen op disk te bewaren en deze terug te lezen. We hebben dit echter niet uitgeprobeerd, omdat deze methode veel trager werkt dan de shellmethode (zie tabellen 1 tot en met 3).

Wanneer u de afbeeldingen en tabellen bekijkt, zal het u opvallen dat de sorteertijden van onze drie drives niet geweldig zijn. Dat klopt, maar op dit moment ken ik geen sorteerroutine die sneller diskbestanden sorteert.

Toch is er nog iets mogelijk om sneller te sorteren, mits het geheugen van de computer dit toelaat:

geheugen > aantal records * recordgrootte

Wanneer we een pointer initialiseren en kijken of het te sorteren databestand in de computer past, dan biedt methode 6 een oplossing. We zijn nu eigenlijk alleen afhankelijk van de snelheid, waarmee de diskdrive eenmalig het gehele bestand kan lezen en schrijven. De floppydisk leest en schrijft constant, waardoor er geen diskpauzes ontstaan, die de tijd nadelig be nvloeden. Tijdens het lezen wordt ieder ingelezen record in het virtuele geheugen opgeslagen en de sorteervolgorde wordt bepaald door alleen de adressen aan te passen van de pointers naar het volgende record, dat in het geheugen aanwezig is. Wanneer alle records gelezen zijn, kunnen ze onmiddellijk teruggeschreven worden naar disk. We zouden eventueel nog groepindelingen kunnen maken, zodat de plaatsbepaling sneller verloopt (kans op diskpauzes nog kleiner). We hoeven niet steeds het geheugenbestand te doorlopen van het eerste naar het laatste record, maar door de voorselectie beperken we het aantal te doorlopen records in het geheugen.

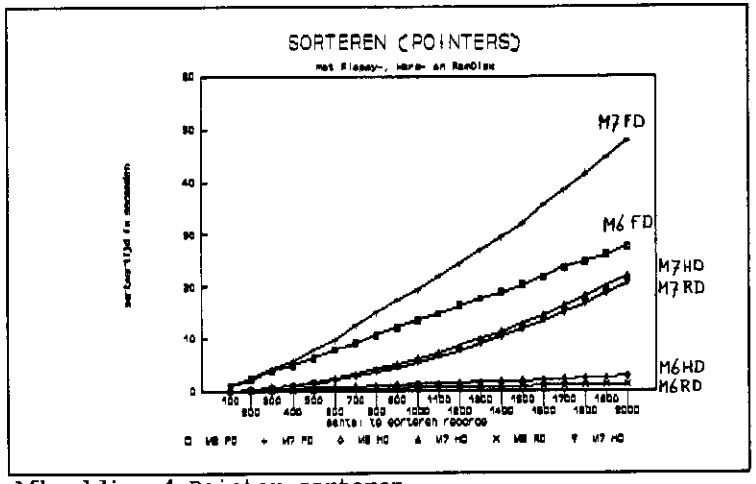


figuur 7
groepindeling

De hierboven gegeven verdeling is willekeurig gekozen. Je zou statistisch moeten bepalen, welk verband de aantallen lettercategorie n hebben. Een simpele verdeling is die van de telefoonklapper, die de namen beginnende met de letters AB, CD, EF, GH, IJ, KL, MN, OP, QR, ST, UVW, XYZ op een tabkaart plaatst. Verdelen we ook teksten die beginnen met een cijfer, bijvoorbeeld 01, 23, 45, 67 en 89, dan kan de rest in een bulkindex geplaatst worden (anders dan alle voorgaande indicatie). We hebben nu 18 indicaties en indien de verdeling gelijkmatig is, zitten er in iedere index ongeveer 120 records van het totale bestand van 2000 records, en dat zal een kleine reductie opleveren van de totale sorteertijd. Dit is niet precies uit de tabellen te halen en zal dus uitgeprobeerd moeten worden, indien u deze verdeling wilt toepassen.

Methode 7 werkt niet met een enkele pointer maar met een array waarvan alle velden van het type pointer zijn. We moeten nu wel weten bij het compileren, hoeveel records we maximaal moeten kunnen sorteren, en of alle records uit het bestand in de computer passen. Wanneer alle data van disk gelezen zijn, begint het eigenlijke sorteren van het array via de recursieve methode. Deze werkt in deze toepassing perfect. Is het bestand in het geheugen gesorteerd, dan moet ook hier het bestand in zijn geheel naar disk worden teruggeschreven.

Wanneer we de tijden van deze twee methoden met elkaar vergelijken (afbeelding 4), blijkt dat methode 6, zonder enige twijfel of voorbehoud, de snelste sorteermethode is.



Afbeelding 4 Pointer sorteren.

Het lijkt dus verstandig om in uw computerprogramma een voorwaarde te formuleren, die bepaalt met welke methode uw programma gaat sorteren.

Als Vrij geheugen > Grootte Databestand	
Ja	Nee
Lees bestand Sorteer bestand Herschrijf bestand	Sorteer via diskdrive

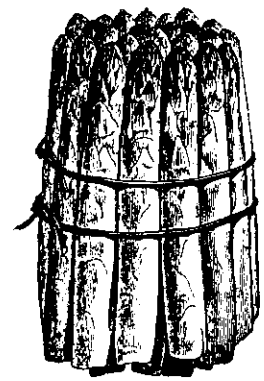
figuur 8. keuze van methode

Tot zover de basis van de sorteerroutines. We hebben in de schema's de termen LeesRecord(), SchrijfRecord() en WisselRecords(,) zien staan die niet verder gedefinieerd zijn. Deze Record()-aanduidingen roepen procedures aan, die bepaalde specifieke taken voor de routines uitvoeren. In het definitieve programma moeten deze dus gedefinieerd worden, alsmede het openen en sluiten van de datafile. Wat we ook nog nodig hebben, is een databestand met te sorteren gegevens.

Ik heb hiervoor een aflopend (van Z tot A) gesorteerd bestand genomen, waaruit we steeds het aantal te kopiëren records halen om deze oplopend (van A tot Z) te sorteren. De recordgrootte is 16 tekens (15 tekens en een lengtebyte).

Het sorteren deden we met 8 sorteerroutines die oplopend met een stapgrootte van 100 records in totaal 2000 records moeten sorteren. We hebben daarbij ook nog 3 verschillen type diskdrives gebruikt, namelijk

1 een ram-drive: drive D (RD)
2 een harddisk: drive C (HD)
3 een floppydisk 720k: drive B, 80 tracks, 9 sectoren (FD); deze is redelijk vergelijkbaar met het NewBrainformaat



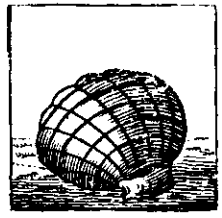
We hebben in het testprogramma enige maatregelen getroffen om de totale tijd enigszins te beperken. Wanneer een sorteermethode twee minuten of meer nodig heeft om een bepaald aantal records te sorteren, dan wordt de procedure afgebroken. Dit is ook de reden, dat ik niet op de NewBrain of de Proton ben gaan testen, maar daarvoor een wat snellere IBM-computer (sorry NewBrainers) heb genomen. De complete test, waarbij iedere sorteerstap van 100 records 7 maal werd uitgevoerd, voor alle methoden, heeft ongeveer 24 uur in beslag genomen. De tijden zijn door de interne klok vastgelegd in een databestandje, waardoor ik rustig op een oor de resultaten kon afwachten . . .

Het kan zijn dat u zegt, ik beschik over veel betere procedures of functies om databestanden te sorteren. In dat geval ben ik, en hopelijk ook andere gebruikers in onze gebruikersgroep, daarin zeer geïnteresseerd, omdat ik al enige jaren naar betere en snellere methoden op zoek ben. Zo had Borland, de uitgever van Turbo Pascal, tot versie 4 een Toolkit-diskette voor het

werken met databestanden. Helaas heeft Borland deze diskette uit de handel genomen en is hij nergens meer te koop, zelfs niet bij Borland in Green Hills Road, U.S.A. Ja, erger nog: niemand zou zo'n diskette hebben, want wie je ook vraagt, het antwoord luidt 'nee, nooit van gehoord of gezien'.
In mei zal ik binnen de cpm/ms-dosgg informeren, of daar sorteerroutines in de bibliotheek aanwezig zijn. Mogelijk hebben de gebruikersgroep of leden daarvan hiervoor ook belangstelling, zodat we misschien in onze bibliotheek voor de NewBrain of Proton geschikte routines kunnen opnemen.

Na het lezen van deze tekst zult u zich misschien afvragen, wat doen we met deze sorteerroutines. Het antwoord is simpel. In de volgende On-line zal ik aan de hand van gesorteerde indexbestanden, gesorteerd via de memory- of shellmethode, eens gaan kijken, hoe we snel een bepaald record kunnen vinden, records invoegen, records verwijderen in een gesorteerde lijst of records verplaatsen, omdat het sorteerveld (keyveld) gewijzigd is.

Bas Boetekees



DE RESULTATEN VAN DE METINGEN

methode 1: normaal sorteren zonder geheugenvariabele
methode 2: normaal sorteren met geheugenvariabele, via diskdrive
methode 3: shellsorteren zonder geheugenvariabele, diskdrive
methode 4: shellsorteren met geheugenvariabele, via diskdrive
methode 5: labelsorteren met geheugenvariabele, via diskdrive
methode 6: pointersorteren met virtueel geheugen
methode 7: recursief sorteren met pointerarray en virtueel geheugen
methode 8: semirecursief sorteren, via diskdrive

TABEL 1: Sorteren over de FLOPPY-DISK.

Demonstratie Sorteerprogramma's met Drive B:								
Methode	1	2	3	4	5	6	7	8
AANTAL	TIJD	TIJD	TIJD	TIJD	TIJD	TIJD	TIJD	TIJD
======	===.==	===.==	===.==	===.==	===.==	===.==	===.==	===.==
100	24.94	47.50	1.90	1.84	1.20	1.08	1.07	1.66
200	128.58	167.18	5.68	5.63	3.19	2.30	2.50	6.33
300	295.77	350.85	10.94	10.99	5.23	3.82	4.22	13.70
400			15.86	15.12	7.96	5.01	5.80	23.63
500			22.37	21.80	9.61	6.38	7.77	36.34
600			29.87	33.90	13.20	7.93	9.60	51.75
700			38.20	36.76	58.43	9.10	12.51	
800			50.11	48.77	139.02	10.75	15.01	
900			64.11	54.77		12.14	17.28	
1000			80.84	79.07		13.52	19.35	
1100			98.04	96.68		14.77	22.05	
1200			109.82	116.43		16.24	24.24	
1300			114.58	123.10		17.65	26.76	
1400			119.95			18.84	29.39	
1500						20.16	32.11	
1600						21.87	35.51	
1700						23.63	38.37	
1800						24.64	41.31	
1900						26.02	44.54	
2000						27.52	47.71	

TABEL 2: Sorteren over de HARD-DISK.

Demonstratie Sorteerprogramma's met Drive C:								
Methode	1	2	3	4	5	6	7	8
AANTAL	TIJD	TIJD	TIJD	TIJD	TIJD	TIJD	TIJD	TIJD
======	===.==	===.==	===.==	===.==	===.==	===.==	===.==	===.==
100	10.34	5.38	0.72	0.57	0.79	0.11	0.16	1.44
200	47.10	23.49	1.96	1.62	1.86	0.25	0.44	5.57
300	110.67	50.43	3.42	3.09	3.03	0.37	0.82	12.38
400	198.75	86.25	4.81	4.13	4.26	0.51	1.29	21.88
500	312.80	133.56	6.51	5.02	5.53	0.65	1.86	34.12
600		187.28	8.29	7.34	6.86	0.78	2.54	49.03
700		253.71	10.53	9.06	11.59	0.91	3.29	69.06
800			12.51	10.91	15.69	1.07	4.19	91.09
900			15.29	12.07	19.45	1.20	5.16	117.30
1000			17.50	14.20	22.00	1.33	6.23	
1100			19.32	17.77	25.78	1.47	7.35	
1200			22.42	20.16	28.21	1.61	8.59	
1300			25.25	22.17	31.24	1.74	9.92	
1400			26.34	23.94	33.99	1.89	11.33	
1500			29.67	25.90	37.61	2.01	12.89	
1600			31.54	27.46	40.67	2.19	14.52	
1700			33.99	29.76	45.20	2.31	16.25	
1800			37.87	30.95	48.47	2.45	18.06	
1900			39.62	33.66	50.30	2.58	19.98	
2000			42.56	34.91	53.76	2.72	21.99	

TABEL 3: Sorteren over de RAM-DISK.

Demonstratie Sorteerprogramma's met Drive D:								
Methode	1	2	3	4	5	6	7	8
AANTAL	TIJD	TIJD	TIJD	TIJD	TIJD	TIJD	TIJD	TIJD
======	===.==	===.==	===.==	===.==	===.==	===.==	===.==	===.==
100	9.68	2.95	0.67	0.54	0.81	0.11	0.11	1.44
200	42.44	11.71	1.69	1.39	1.88	0.13	0.32	5.55
300	101.22	26.28	2.91	2.47	3.07	0.18	0.62	12.39
400	188.92	46.73	4.21	3.40	4.35	0.24	1.02	21.92
500	309.71	72.78	5.89	4.47	5.71	0.31	1.53	34.17
600		104.96	7.24	6.14	7.14	0.37	2.12	49.19
700		142.56	9.05	7.64	8.67	0.42	2.81	67.24
800		186.06	10.58	8.65	10.28	0.48	3.61	88.04
900		236.29	13.01	10.09	11.94	0.55	4.50	111.78
1000		291.29	14.93	11.73	13.61	0.61	5.51	
1100			16.53	14.16	15.27	0.69	6.57	
1200			18.84	16.05	17.06	0.72	7.71	
1300			21.70	18.43	18.77	0.78	8.97	
1400			23.66	20.10	20.65	0.84	10.33	
1500			25.99	21.64	22.40	0.91	11.79	
1600			28.14	23.52	24.43	0.97	13.34	
1700			30.74	25.72	26.36	1.04	14.98	
1800			34.58	27.75	28.47	1.10	16.70	
1900			36.61	30.80	30.29	1.16	18.56	
2000			40.32	32.71	32.52	1.21	20.50	

Kijk eens naar het verschil tussen de harddisk en de ram-disk voor de methoden 2 tot en met 4. Zou een goede harddisk dan (bijna) even snel zijn als relatief duur ram-geheugen!

STANDAARD PROCEDURES EN FUNCTIES

```
Type  SRecord  = Record                    { definieer sorteerrecord }
                Tekst : String[Lengte];     { een record kan uit meerdere }
                                            { velden bestaan, maar dat is }
              End;                          { niet van belang }

Procedure LeesRecord(X: Integer;            { plaats in datafile }
                     Var Sort: SRecord);    { werkrecord }
Begin                                       { record X staat op }
 Seek(FSorteer, X-1 );                      { filepositie  X-1 }
 Read(FSorteer, Sort)                       { lees het werkrecord }
End;                                        { sort bevat de gegevens, }
                                            { die aan de aanroep }
                                            { variabele wordt gegeven }
Procedure SchrijfRecord(X: Integer;         { de plaats in datafile }
                        Sort: SRecord);     { de gegevens }
Begin
 Seek(FSorteer, X-1 );                      { de plaats in de datafile }
 Write(FSorteer, Sort);                     { schrijf de gegevens weg }
End;

Function SortBuf(X:Integer):String;         { geef zgn keyveld als resultaat }
Begin                                       { evt andere velden zijn niet nodig }
 LeesRecord(X, WSorteer);                   { lees recordnummer }
 SortBuf:=WSorteer.Tekst;                   { geef tekstveld als uitvoer }
End;

Procedure WisselRecords(X, Y:Integer);
Var Dummy1, Dummy2: SRecord;
Begin
 LeesRecord(X, Dummy1);   { lees gegevens en bewaar deze in geheugen }
 LeesRecord(Y, Dummy2);   { lees gegevens op de 2e opgegeven plaats }
 SchrijfRecord(X, Dummy2);  { schrijf deze op de 1e opgegeven plaats }
 SchrijfRecord(Y, Dummy1); { schrijf dummy op de 2e opgegeven plaats }
End;                       { inhoud van plaats 1 en 2 zijn verwisseld }
```

SORTEERPROCEDURES

METHODE 1 EN 2: NORMAAL OF BUBBLE SORTEREN

```
Procedure NormSorteer(Aantal: Integer;          { te sorteren records }
                      Mode: Char;       { A = ascending, D = descending }
                      Memory: Boolean);  { bewaar records in geheugen }

Var I, J: Integer;                  { tellers voor positie I en J records }
    IRecord: SRecord;          { inhoud van het I-de record in geheugen }
    JRecord: SRecord;          { inhoud van het J-de record in geheugen }
    Gewisseld: Boolean;  { inhoud I-de record is gewisseld in geheugen }
    Verander: Boolean;          { records moeten gewisseld worden }
Begin
 Mode:=Upcase(Mode);                { gebruik altijd hoofdletters }
 For I:=1 To Aantal-1 Do
 Begin
  If Memory                         { gebruik ook het geheugen }
  then Begin
       LeesRecord(I, IRecord);      { plaats record I in IRecord }
       Gewisseld:=False;            { geen wisseling uitgevoerd }
```

```
      End;
   Verander:=False;                          { geen verandering van records }
   For J:= I+1 To Aantal Do
   Begin
    If Memory                                  { gebruik ook het geheugen }
    then Begin
         LeesRecord(J, JRecord);             { plaats record J in JRecord }
         Case Mode Of                                   { sorteermethode }
          'A': If IRecord.Tekst > JRecord.Tekst                { oplopend }
               then Verander:=True;
           else If IRecord.Tekst < JRecord.Tekst               { aflopend }
               then Verander:=True
         End
        End
   else Case Mode Of                                    { sorteermethode }
         'A': If SortBuf(I)>SortBuf(J)                         { oplopend }
              then Verander:=True;
         else If SortBuf(I)<SortBuf(J)                        { aflopend }
              then Verander:=True
        End;
   If Verander                                  { inhoud records veranderen }
   then Case Memory Of                                { gebruik geheugen }
         True: Begin                                                { ja, }
                SchrijfRecord(J, IRecord);          { IRecord naar plaats J }
                IRecord:=JRecord;       { bewaar oude JRecord in IRecord }
                Gewisseld:=True;                    { records zijn gewisseld }
               End;
         else  WisselRecords(I, J);                  { verwissel de records }
        End;
  End;
  If(Memory) And(Gewisseld)                  { er is gewisseld, dan MOET }
  then SchrijfRecord(I, IRecord)          { IRecord weggeschreven worden }
 End;
End;
```

METHODE 3 EN 4: SHELLSORTEREN

```
Procedure ShellSorteer(Aantal: Integer;       { aantal te sorteren records }
                       Mode: Char    );             { A = ascending sorteren }
                            { Memory: Boolean }{ wordt hier niet gebruikt }
Var I, J, K, L: Integer;
   Afstand: Integer;
```



```
    JRecord: SRecord;
    LRecord: SRecord;
    Verander: Boolean;
    Gewisseld: Boolean;
Begin
 Afstand:= Aantal Div 2;                          { bepaal sorteerafstand }
 While Afstand > 0 Do                          { als afstand is groter dan 0 }
 Begin
  For I:= Afstand+1 To Aantal Do
  Begin
   J:= I - Afstand;
   While J > 0 Do
   Begin
    Verander:=False;                          { geen records verwisselen }
    L:= J + Afstand;
    LeesRecord(J, JRecord);                                            { }
    LeesRecord(L, LRecord);       { het record wordt eenmaal gelezen }
     'A': If JRecord.tekst > LRecord.Tekst
          then Verander:=True                  { wissel inhoud op J en L }
          else J:=0;
     else If JRecord.Tekst < LRecord.Tekst
          then Verander:=True
          else J:=0;
    End;
    If Verander                                { er moet gewisseld worden }
    then Begin
          SchrijfRecord(J, LRecord);                { overschrijf J met L }
          SchrijfRecord(L, JRecord);                              { L met J }
         End
    J:= J - Afstand
   End;
  End;
  Afstand:= Afstand Div 2;
 End;
End;
```

METHODE 5: LABELSORTEREN

```
Procedure LabelSorteer(Aantal: Integer;      { aantal te sorteren records }
                       Mode: Char    );                    { A = oplopend }
                       Memory: Boolean);        { wordt hier niet gebruikt }
Label H1, H2, H3, H4, H5   , H7, H8, H9, H0;
```

```
Var I, J, L, R: Integer;
    IRecord: SRecord;                { record op plaats I in databestand }
    JRecord: SRecord;                { record op plaats J in databestand }
    LRecord: SRecord;                { record op plaats L in databestand }
    WRecord: SRecord;                { record op plaats ? in databestand }
    Wissel: Boolean;
Begin
 Mode:=Upcase(Mode);                 { zet mode om in hoofdletters }
 H1: L:=Aantal div 2 + 1;                        { bepaal het midden }
    R:=Aantal;                        { laatste te sorteren record }
 H2: If L>1                          { als L is niet het eerste record }
    then Begin
         L:=L-1;
         LeesRecord(L, LRecord); { lees record op plaats L in LRecord }
        End
    else Begin
         LeesRecord(R, LRecord);      { lees laatste record in LRecord }
         LeesRecord(1, WSorteer );    { lees eerste record in WRecord }
         SchrijfRecord(R, WSorteer ); { en plaats hem op laatste plaats }
         R:=R-1;
         If R=1
         then Begin
              SchrijfRecord(1, LRecord);
              Goto H0;                { het bestand is gesorteerd en }
             End;                       { verlaat dus de procedure }
        End;
 H3: J:=L;
 H4: I:=J;
    J:=J+J;
    If J > R then goto H8;
    If J = R then goto H7;
 H5: Case Mode Of
     'A': If SortBuf(J) < SortBuf(J+1) then J:=J+1;
     else If SortBuf(J) > SortBuf(J+1) then J:=J+1;
    End;
 H7: LeesRecord(J, JRecord);
    SchrijfRecord(I, JRecord);
    Goto H4;
 H8: J:=I;
    I:=J Div 2;
 H9: If(J=L)
    then Wissel:=True
    else Begin
```


```
         LeesRecord(I, IRecord);
         Case Mode Of
          'A': If LRecord.Tekst<=IRecord.Tekst
               then Wissel:=True else Wissel:=False;
          else If LRecord.Tekst>=IRecord.Tekst
               then Wissel:=True else Wissel:=False;
         End;
        End;
    If Wissel                        { de inhoud van LRecord wegschrijven }
    then Begin
         SchrijfRecord(J, LRecord);
         Goto H2;
        End
    else Begin                       { de inhoud van IRecord wegschrijven }
         SchrijfRecord(J, IRecord);
         Goto H8;
        End;
 H0: Begin End;
End;
```

METHODE 6: LEES DATA EN SORTEER VIA POINTER

```
Procedure MemorySorteer1(Aantal:Integer);
Type RecPtr  = ^PtrRec;
    PtrRec  = Record
              Inhoud: String[Lengte];
              Volgende: RecPtr;
             End;
Var  R: LongInt;
    H, V, Ptr: RecPtr;

 Procedure Sorteer_Volgende(Var DitRecord:SRecord);
 Begin
  If Ptr=Nil
  then
   Begin                                  { dit is het eerste werkpakket }
    New(Ptr);
    Ptr^.Inhoud:=DitRecord.Tekst;
    Ptr^.Volgende:=Nil;
   End
  else
   Begin                                  { kijk of werkpakket+nr bestaat }
```

```
      H:= Ptr;                                    { huidige pointerpositie }
      Repeat
       V:= H;                                        { vorige is huidige }
       H:= H^.Volgende;                                    { volgende }
      Until(H^.Inhoud > DitRecord.Tekst) Or(H = Nil);
      If Ptr^.Inhoud > DitRecord.Tekst
      then                                        { de eerste pointerinhoud }
        Begin                                     { is groter dan deze regel }
         New(Ptr);                                { dit wordt dus de eerste }
         PTr^.Volgende:=V;                              { volgend is vorige }
         Ptr^.Inhoud:=DitRecord.Tekst;                   { kopieer werkrec }
        End
      else
       Begin                                  { regel komt ergens in de lijst }
        New(V^.Volgende);                        { nieuwe regel voor vorige }
        V^.Volgende^.Volgende:=H;              { nieuw record wordt huidige }
        V^.Volgende^.Inhoud:=DitRecord.Tekst       { kopieer werkrecord }
       End;
     End;
   End;

Begin
 R:=1;
 Ptr:=Nil;
 For R:=1 to Aantal Do
 Begin
  LeesRecord(R, DitRecord);                        { lees sortrecord }
  Sorteer_Volgende(DitRecord);                     { bepaal zijn plaats }
 End;
 R:=1;                                             { reset de filewijzer }
 H:=Ptr;                                               { eerste pointer }
 While H<>Nil Do
 Begin
  DitRecord.Tekst:=H^.Inhoud;
  SchrijfRecord(R, DitRecord);                     { schrijf sortrecord }
  H:=H^.Volgende;                                        { volgende }
  Inc(R);                                          { verhoog filewijzer }
 End;
 H:=Ptr;
 While Ptr<>Nil Do                                 { verwijder pointers en }
 Begin                                               { geheugenruimte vrij }
  H:=Ptr^.Volgende;                              { volgende record in mem }
  Dispose(Ptr);                                   { maak deze ruimte vrij }
```



```
  Ptr:=H;                                                { dit is nu de eerste }
 End;
End;
```

METHODE 7: POINTER ARRAY

```
Procedure MemorySorteer1(Aantal:Integer);
Type RecPtr  = ^PtrRec;
     PtrRec  = Record
                Inhoud: String[Lengte];
                Volgende: RecPtr;
               End;
Var  R: LongInt;
     H, V, Ptr: RecPtr;

  Procedure Sorteer_Volgende(Var DitRecord:SRecord);
  Begin
   If Ptr=Nil
   then
    Begin                                  { dit is het eerste werkpakket }
     New(Ptr);                                   { creeer de eerste entry }
     Ptr^.Inhoud:=DitRecord.Tekst;
     Ptr^.Volgende:=Nil;                { er is geen volgend pointerrecord }
    End
   else
    Begin                                { kijk of werkpakket+nr bestaat }
     H:= Ptr;                                    { huidige pointerpositie }
      Repeat
       V:= H;                                        { vorige is huidige }
       H:= H^.Volgende;                        { volgende pointerrecord }
      Until(H^.Inhoud > DitRecord.Tekst) Or(H = Nil);
      If Ptr^.Inhoud > DitRecord.Tekst
      then                                        { de eerste pointerinhoud }
       Begin                                    { is groter dan deze regel }
        New(Ptr);                               { dit wordt dus de eerste }
        PTr^.Volgende:=V;                             { volgend is vorige }
        Ptr^.Inhoud:=DitRecord.Tekst;                 { kopieer werkrecord }
       End
      else
       Begin                                { regel komt ergens in de lijst }
        New(V^.Volgende);                       { nieuwe regel voor vorige }
        V^.Volgende^.Volgende:=H;              { nieuw record wordt huidige }
```

```
      V^.Volgende^.Inhoud:=DitRecord.Tekst        { kopieer werkrecord }
     End;
   End;
 End;

Begin
 R:=1;                                          { teller naar 1e record }
 Ptr:=Nil;                                        { zet pointer op Nil }
 For R:=1 to Aantal Do                          { haal alle records uit file }
 Begin
  LeesRecord(R, DitRecord);                          { lees sortrecord }
  Sorteer_Volgende(DitRecord);                     { bepaal zijn plaats }
 End;
 R:=1;                                          { reset de filewijzer }
 H:=Ptr;                                            { eerste pointer }
 While H<>Nil Do
 Begin
  DitRecord.Tekst:=H^.Inhoud;
  SchrijfRecord(R, DitRecord);                       { schrijf sortrecord }
  H:=H^.Volgende;                                              { volgende }
  Inc(R);                                          { verhoog filewijzer }
 End;
 H:=Ptr;
 While Ptr<>Nil Do
 Begin
  H:=Ptr^.Volgende;                          { volgende record in geheugen }
  Dispose(Ptr);                                  { maak deze ruimte vrij }
  Ptr:=H;                                          { dit is nu de eerste }
 End;
End;
```

Methode 8: Semi QuickSorteer.

```
Procedure SemiRecursief(Aantal:Integer; Mode:Char; Memory:Boolean);
Type Ptr_Type = ^Type_Ptr;
     Type_Ptr = Record
                 Laag: Integer;
                 Hoog: Integer;
                 I: Integer;
                 Punt: Byte;
                 Volg: Ptr_Type;
                End;
Var  Ptr, Hulp: Ptr_Type;
```



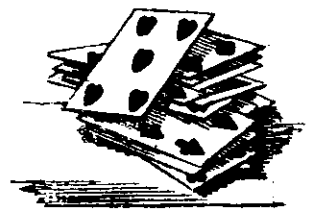
```
Procedure PushWaarden(Var Laag, Hoog, I:Integer; P:Byte);
Begin
 If Ptr=Nil
 then Begin
       New(Ptr);                                { maak eerste pointerrecord }
       Ptr^.Volg:=Nil;                          { er zijn geen records meer }
      End
 else Begin
       Hulp:=Ptr;                                  { bewaar oude recordpointer }
       New(Ptr);                  { maak een nieuwe en reserveer de ruimte }
       Ptr^.Volg:=Hulp;                      { verbind de nieuwe aan de oude }
      End;
 Ptr^.Laag:=Laag;                  { sla de variabelen op in het geheugen }
 Ptr^.Hoog:=Hoog;
 Ptr^.I:=I;
 Ptr^.Punt:=P;                  { dit is de plaats waarvan gePUSHed werd }
End;

Procedure PopWaarden(Var Laag, Hoog, I:Integer);
Begin
 Laag:=Ptr^.Laag;                      { geef de opgeslagen waarden terug }
 Hoog:=Ptr^.Hoog;                                        { aan de procedure }
 I:=Ptr^.I;
 Hulp:=Ptr^.Volg;                  { bewaar plaats volgende pointerrecord }
 Dispose(Ptr);                       { verwijder top en maak ruimte vrij }
 Ptr:=Hulp;                         { dit is de vorige gePUSHde aanroep }
End;

Procedure Sorteer(Laag, Hoog:Integer);
Label H0, H1, H2, H3, H4, H5;
Var I, J: Integer;
    Spil: SRecord;
Begin
 H0: If Laag < Hoog
     then
      Begin
       I:=Laag;
       J:=Hoog;
       LeesRecord(J, Spil);
       Repeat
        While(I<J) And(SortBuf(I)<=Spil.Tekst) Do Inc(I);
        While(J>I) And(SortBuf(J)>=Spil.Tekst) Do Dec(J);
        If I<J then WisselRecords(I, J);
```

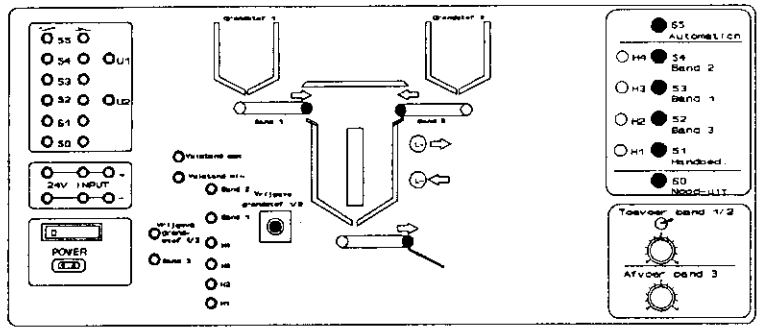
```
      Until I>=J;
      WisselRecords(I, Hoog);
      IF(I-Laag) <(Hoog-I)
      then
       Begin
        PushWaarden(Laag, Hoog, I, I);     { sla op var Laag, Hoog en I }
                    {   Sorteer(Laag, I-1) }{ sorteer de onderste helft }
        Hoog:=I-1;                                        { stel Hoog in }
        Goto H0;                              { hier moeten we terugkeren }
H1:                 { Sorteer(I+1, Hoog); }{ sorteer de bovenste helft }
        Laag:=Ptr^.I+1;
        Hoog:=Ptr^.Hoog;
        Ptr^.Punt:=2;                         { De uitgang wordt label H2 }
        Goto H0;
H2:     End                                          { verlaat de procedure }
      else
       Begin
        PushWaarden(Laag, Hoog, I, 3);     { sla op var Laag, Hoog en I }
                    { Sorteer(I+1, Hoog); }{ sorteer de bovenste helft }
        Laag:=I+1;
        Goto H0;                              { hier moeten we terugkeren }
H3:                  { Sorteer(Laag, I-1); }{ sorteer de onderste helft }
        Laag:=Ptr^.Laag;
        Hoog:=Ptr^.I-1;
        Ptr^.Punt:=4;                         { De uitgang wordt label H4 }
        Goto H0;
H4:     End                                          { verlaat de procedure }
      End;
H5:  If Ptr<>Nil
      then Begin
           Case Ptr^.Punt Of
            1: Goto H1;
            3: Goto H3;
           End;
           PopWaarden(Laag, Hoog, I);
           Goto H5;
          End;
End;

Begin
 Ptr:=Nil;
 Sorteer(1, Aantal);
End;
```



meng-installatie

In de uitbreiding van het Nieuwe Technologieën Lokaal van Streekschool Leiden is weer een hoeveelheid nieuwe apparatuur binnengekomen. Bij deze apparatuur waren ook twee procesmodellen van het merk ELWE. Hiermee kunnen verschillende processen gesimuleerd worden. Gelijk kwam de gedachte bij mij op, dat deze procesmodellen uitstekend geschikt zijn om te gebruiken in combinatie met de NewBrain, de buitenwereldkaart en de i/o-kaart. Als eerste heb ik het procesmodel uitgevoerd als menginstallatie. Hieronder is een schets van het procesmodel weergegeven.



PROCESMODEL

1. Het linkergedeelte van het procesmodel dient voor in- en uitvoer van signalen.
2. De rechterzijde van het procesmodel dient om het proces te bedienen.
3. Het middelste gedeelte van het procesmodel dient om het gesimuleerde proces weer te geven.

Het procesmodel biedt een aantal mogelijkheden die hieronder zijn aangegeven.

1. Handbediening.

Met S1 kan het procesmodel op handbediening worden gezet.

1a. Met drukknop S3 kan men de silo met grondstof 1 via band 1 vullen.
1b. Met drukknop S4 kan men de silo met grondstof 2 via band 2 vullen.
1c. Met drukknop S2 kan het mengsel via band 3 uit de silo afgevoerd worden.

Het vullen kan slechts tot de hoogte L+. Is dit punt bereikt, dan wordt het vulproces afgebroken. Het vullen van de silo wordt gesimuleerd met een bargraph.
Het afvoeren van het mengsel kan slechts tot L-. Als dit punt bereikt is, stopt het afvoeren van het mengsel.
Het proces wordt gestopt als Nood-uit of Automatisch wordt bediend.

2. Automatisch.

In de stand automatisch begint het proces met het vullen van de silo via band 1 en band 2. Dit wordt ook weer beveiligd door L+, waarna het afvoeren wordt gestart.
Het afvoeren van het mengsel via band 3 wordt gestopt als L- is bereikt, waarna het vullen wordt gestart.
Het proces wordt gestopt als Nood-uit of Handbediening wordt bediend.

Via Nood-uit kan men het programma beëindigen.

Maarten Floor

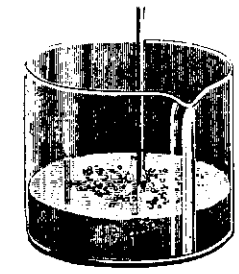
BASICPROGRAMMA MENGINSTALLATIE

```
10   FOR t = 1 TO 255 : CLOSE #t : NEXT t
20   OPEN #0, 0 : PUT 23, 66 : OPEN #6, 6
30   FOR t = 112 TO 123 : OPEN #t, 7, t : NEXT t : CLOSE #104 :
     OPEN #104, 7, 104
40   PUT #122, 207, 0 : REM init pioA output
50   PUT #120, 0 : pa = 0 : REM reset output A
60   CLOSE #6 : OPEN #6, 6 : REM input toetsenbord
70   REM Schermopbouw
80   PUT 22, 2, 1 : ? "M E N G I N S T A L L A T I E"
90   PUT 22, 5, 4 : ? "AUTOMATISCH"
100  PUT 22, 5, 6 : ? "Band 2"
110  PUT 22, 5, 8 : ? "Band 1"
```

```
120  PUT 22, 5, 10 : ? "Band 3"
130  PUT 22, 5, 12 : ? "HANDBEDIENING"
140  PUT 22, 5, 14 : ? "NOOD-UIT"
150  GOTO 680
160  GOSUB 970
170  IF a = 134 THEN mi = 1 : ma = 0
180  IF a = 142 THEN mi = 1 : ma = 0
190  IF a = 70 THEN ma = 1 : mi = 0
200  IF a = 198 THEN ma = 1 : mi = 0
210  IF a = 102 THEN ma = 1 : mi = 0
220  IF a = 86 THEN ma = 1 : mi = 0
230  IF a = 118 THEN ma = 1 : mi = 0
240  IF a = 6 THEN b1 = 0 : b2 = 0 : b3 = 0
250  IF a = 134 THEN b1 = 0 : b2 = 0 : b3 = 0
260  IF a = 150 THEN b1 = 1 : b2 = 0 : b3 = 0
270  IF a = 22 THEN b1 = 1 : b2 = 0 : b3 = 0
280  IF a = 38 THEN b1 = 0 : b2 = 1 : b3 = 0
290  IF a = 166 THEN b1 = 0 : b2 = 1 : b3 = 0
300  IF a = 14 THEN b1 = 0 : b2 = 0 : b3 = 1
310  IF a = 142 THEN b1 = 0 : b2 = 0 : b3 = 1
320  IF a = 206 THEN b1 = 0 : b2 = 0 : b3 = 1
330  IF a = 78 THEN b1 = 0 : b2 = 0 : b3 = 1
340  IF a = 6 THEN ma = 0 : mi = 0
350  IF a = 102 THEN ma = 1 : mi = 0
360  IF a = 86 THEN ma = 1 : mi = 0
370  IF a = 118 THEN ma = 1 : mi = 0
380  IF a = 4 THEN ha = 1 : au = 0
390  IF a = 68 THEN ha = 1 : au = 0
400  IF a = 132 THEN ha = 1 : au = 0
410  IF a = 196 THEN ha = 1 : au = 0
420  IF a = 2 THEN ha = 0 : au = 1
430  IF a = 66 THEN ha = 0 : au = 1
440  IF a = 130 THEN ha = 0 : au = 1
450  IF a = 194 THEN ha = 0 : au = 1
460  IF ha = 1 AND b1 = 1 AND ma = 0 THEN PUT 22, 1, 8, 49 : PUT
     #120, 149
470  IF ha = 1 AND b1 = 1 AND ma = 1 THEN PUT 22, 1, 8, 48 : PUT
     #120, 1
480  IF b1 = 0 THEN PUT 22, 1, 8, 48
490  IF ha = 1 AND b2 = 1 AND ma = 0 THEN PUT 22, 1, 6, 49 : PUT
     #120, 153
500  IF ha = 1 AND b2 = 1 AND ma = 1 THEN PUT 22, 1, 6, 48 : PUT
     #120, 1
```



```
510  IF b2 = 0 THEN PUT 22, 1, 6, 48
520  IF ha = 1 AND b3 = 1 AND mi = 0 THEN PUT 22, 1, 10, 49 : PUT
     #120, 69
530  IF ha = 1 AND b3 = 1 AND mi = 1 THEN PUT 22, 1, 10, 48 : PUT
     #120, 1
540  IF b3 = 0 THEN PUT 22, 1, 10, 48
550  IF ha = 1 THEN PUT 22, 1, 12, 49
560  IF ha = 0 THEN PUT 22, 1, 12, 48
570  IF au = 1 THEN PUT 22, 1, 4, 49 : b1 = 0 : b2 = 0 : b3 = 0 : ha =
     0
580  IF au = 0 THEN PUT 22, 1, 4, 48
590  IF a = 199 THEN GOTO 670
600  IF a = 7 THEN GOTO 670
610  IF a = 135 THEN GOTO 670
620  IF a = 71 THEN GOTO 670
630  IF ha = 1 AND a = 6 THEN PUT #120, 1
640  IF ha = 1 AND a = 68 THEN PUT #120, 1
650  IF au = 1 THEN PUT #120, 0 : GOTO 840
660  GOTO 160
670  PUT 22, 5, 18 : ? "DE INSTALLATIE IS GESTOPT"
680  PUT 22, 1, 4, 48, 22, 1, 6, 48, 22, 1, 8, 48, 22, 1, 12, 48, , 22, 5,
     20 : ? "KIES HANDBEDIENING, AUTOMATISCH OF" : PUT 22, 5,
     22 : ? "SPATIE OM TE STOPPEN."
690  au = 0 : ha = 0 : PUT #120, 0
700  GET #6, b
710  GET #104, a
720  IF b = 32 THEN PUT 31 : END
730  IF a = 2 THEN au = 1 : GOTO 820
740  IF a = 130 THEN au = 1 : GOTO 820
750  IF a = 66 THEN au = 1 : GOTO 820
760  IF a = 194 THEN au = 1 : GOTO 820
770  IF a = 4 THEN ha = 1 : GOTO 820
780  IF a = 132 THEN ha = 1 : GOTO 820
790  IF a = 68 THEN ha = 1 : GOTO 820
800  IF a = 196 THEN ha = 1 : GOTO 820
810  GOTO 700
820  PUT 22, 1, 18, 30, 22, 1, 20, 30, 22, 1, 22, 30
830  GOTO 160
840  IF ma = 0 THEN PUT #120, 156
850  IF ma = 1 THEN PUT #120, 68
860  IF a = 134 THEN PUT #120, 156
870  GOSUB 970
880  IF a = 70 THEN mi = 0 : PUT #120, 68
```



```
890   IF a = 134 THEN ma = 0 : PUT #120, 156
900   IF a = 71 THEN PUT #120, 0 : GOTO 670
910   IF a = 135 THEN PUT #120, 0 : GOTO 670
920   IF a = 7 THEN PUT #120, 0 : GOTO 670
930   IF a = 4 THEN PUT #120, 1 : GOTO 1020
940   IF a = 68 THEN PUT #120, 1 : GOTO 1020
950   IF a = 132 THEN PUT #120, 1 : GOTO 1020
960   GOTO 870
970   a1 = a
980   GET #104, a
990   IF a1 <> a THEN GOTO 1010
1000  GOTO 980
1010  RETURN
1020  ha = 1 : au = 0 : GOTO 160
```

UITLEG BIJ HET PROGRAMMA

regel 10 - 60	installatie van de poorten van de buitenwereldkaart.
regel 70 - 140	schermopbouw.
regel 170 - 450	bepalen van stuurcodes vanuit het ingangssignaal van de buitenwereldkaart.
regel 460 - 650	sturen van signalen naar de buitenwereldkaart en het scherm.
regel 670 - 680	schermsturingen bij de start van het programma en na het bedienen van nood-uit.
regel 700 - 710	afvangen van signalen van het toetsenbord en de buitenwereldkaart.
regel 720 - 800	bepaling of het programma wordt beëindigd of dat het programma op automatisch of met de hand kan worden bediend.
regel 820	schoonvegen van regels op het scherm.
regel 840 - 860	sturing van de i/o-kaart in de stand automatisch.
regel 880 - 950	sturen van signalen naar de buitenwereldkaart.
regel 980 - 1000	afvangen van signalen vanuit de buitenwereldkaart.

ledenlijst

1007 EA	amsterdam	peter van groen
1011 AZ	amsterdam	k t bakker
1012 MK	amsterdam	mw h tusveld
1013 NJ	amsterdam	l sluis
1015 EA	amsterdam	paul foeken
1033 NM	amsterdam	f tammer
1054 LR	amsterdam	j n van baalen
1055 JA	amsterdam	drenthe
1057 PG	amsterdam	c van der vlies
1061 CH	amsterdam	o strijdhaftig
1065 BH	amsterdam	h g dekker
1074 AD	amsterdam	a kulhan
1077 BR	amsterdam	w van linden tol
1079 WR	amsterdam	frank van schuilenburg
1082 LE	amsterdam	r r blanken
1094 CZ	amsterdam	n stikker
1094 KK	amsterdam	c klijn
1094 LK	amsterdam	j w van hoek
1097 AS	amsterdam	menno stevens
1098 HS	amsterdam	r van albada
1098 KL	amsterdam	w a van hoek
1102 AA	amsterdam zo	a f zwartjes

a elfferich	1121 CP landsmeer
h h l vd woude	1121 CP landsmeer
j w derksen	1191 AK ouderkerk amstel
m a floor	1191 HC ouderkerk amstel
a hekman	1211 KX hilversum
a a mosca roman	1214 AV hilversum
c wielemaker	1214 CP hilversum
mw m j f verhaart	1218 CN hilversum
a von morgen	1231 AP loosdrecht
m w j van harmelen	1241 LV kortenhoef
het spanningveld	1324 CR almere
g fisser	1324 MC almere
t uijldert	1353 CH almere
k steunebrink	1381 AJ weesp
n h j ballast	1399 GK muiderberg
p meyer	1401 SR bussum
m s vreedenburg	1403 GA bussum
simon de bruin	1411 VW naarden
gerrits	1421 XK uithoorn
j c l ditzel	1422 EK uithoorn
g denneman	1521 TW wormerveer
h van berkel	1541 GV koog aan de zaan
a s p kok	1688 DA nibbixwoud
john van der ploeg	1782 AT den helder
c m weel	1782 GJ den helder
r h koolstra	1784 AG den helder
f c a de wit	1788 NX den helder
a w ooyevaar	1827 JA alkmaar
n c j van paassen	1827 NA alkmaar
s a nijhuis	1945 PT beverwijk
b van der zwaag	2036 CM haarlem
mw m h doorduin	2101 HC heemstede
techn bur delta-t	2105 SM heemstede

c hoogland	2105 XV heemstede
r s van der wijk	2153 CE nieuw-vennep
j p koning	2161 RV lisse
koos van touw	2162 HK lisse
j m s zwarts	2201 VK noordwijk zh
sipke de wal	2201 WG noordwijk
maurice hiep	2241 PC wassenaar
p w m van dijk	2241 SB wassenaar
h g j veldman	2263 VX leidschendam
g p eijkhoff	2264 DP leidschendam
p staring	2271 RA voorburg
ronald diepenbroek	2272 EL voorburg
j a c g vd valk	2273 HS voorburg
hasrun iskandar	2274 VH voorburg
r koswal	2280 AD rijswijk
m leger	2282 HL rijswijk zh
t huizenga	2285 HK rijswijk zh
j h c van haarlem	2314 VS leiden
david kroeders	2315 JN leiden
c a van de repe	2316 HJ leiden
j van der burgt	2317 CM leiden
c w jouvenaar	2317 LH leiden
w f speekman	2343 DV oegstgeest
a j j verbaken	2352 JM leiderdorp
bisschop bekkersschool	2402 BT alphen ad rijn
t g m schrama	2465 BB rijnsatenwoude
r bakker	2481 XC woubrugge
f w m pallada	2517 XT 's-gravenhage
a j ladenberg	2551 EM 's-gravenhage
j g j van der linden	2555 ZK 's-gravenhage
davy hessing	2564 EL 's-gravenhage
g j hendriks	2564 SR 's-gravenhage
n vd zwan	2584 RB 's-gravenhage

c vd driessche	2593 CW 's-gravenhage
haags montessorilyc	2596 AK 's-gravenhage
m c meijer	2624 LC delft
p b davis	2624 PS delft
a dubbelt	2628 AM delft
e r silooy	2715 VN zoetermeer
m oosterink	2716 BN zoetermeer
f van der hoeven	2722 BK zoetermeer
b dunnewold	2724 PV zoetermeer
leo w zonneveld	2811 AC reeuwijk
a vd zwaan	2811 AN reeuwijk
gertjan steketee	3065 HA rotterdam
j c wubben	3067 EZ rotterdam
a g m van de laar	3069 BK rotterdam
w r van der ent	3076 CT rotterdam
h m van 't hull	3077 BE rotterdam
j h granville	3078 HD rotterdam
j ter horst	3181 RC rozenburg
willem tak	3201 GH spijkenisse
l etman	3271 XC mijnsheerenland
r vd straten	3311 CS dordrecht
j c w aarnoudse	3314 AP dordrecht
j den bleker	3448 XH woerden
h van der pol	3481 VP harmelen
d de rijke	3512 XJ utrecht
g j w wiselius	3571 XD utrecht
t kreuger	3603 GA maarssen
has boetekees	3607 GS maarssen
kees wijgerde	3705 ZE zeist
jacob scholten	3705 ZL zeist
p w boterenbrood	3742 AL baarn
r j jansen	3742 XX baarn
j van bruggen	3811 HT amersfoort

w m hauptmeijer	3816 AW amersfoort
j j van eekelen	3822 AD amersfoort
jan-adam breukel	3828 RJ hoogland
f de roo	3831 AX leusden
h stolman	3902 GJ veenendaal
w h arnold	3911 AX rhenen
th m kamperman	3972 EG driebergen rijsenb
h j stappershoef	4184 EV opijnen
e j van doesburg	4254 BT sleeuwijk
jaap bant	4266 EL eethen
i van kampen	4414 RS waarde
w m luijt	4461 NH goes
n d van de velde	4475 AG wilhelminadorp
j van eekelen	4611 JX bergen op zoom
g bulthe	4645 EZ putte
h a g m perdaems	4724 AJ wouw
th van lottum	4817 VB breda
m j aarts	4901 AV oosterhout nb
r j n m de bont	5012 GT tilburg
j j bots	5017 HB tilburg
m f j a vd velden	5051 BT goirle
m p j kerstholt	5345 DE oss
a l beerensteyn	5463 NW veghel nb
j j 't hart	5501 HC veldhoven
h b haanstra	5550 AS valkenswaard
j g h e bierings	5552 RB valkenswaard
e p naburgh	5612 PL eindhoven
h velthuys	5632 LA eindhoven
j van heummen	5706 DM helmond
g lemmen	5712 BR someren
h j a steinmann	5914 VR venlo
j j peters	5924 EA venlo
peter c tunissen	6001 BS weert

h e c g roijen	6241 NK	bunde
a j m peeters	6325 BN	berg en terblijt
a a de waart	6501 BC	nijmegen
l l h pijnappel	6531 RL	nijmegen
r kranenburg	6532 TN	nijmegen
g t vroegindeweij	6535 TT	nijmegen
d j hoffman	6561 EX	groesbeek
j a kruize	6601 BD	wijchen
p kramer	6661 BN	elst gld
b zuidema	6706 GD	wageningen
j c j kuiken	6865 BH	doorwerth
j p van doorn	6921 KT	duiven
j hermans	6983 ES	doesburg
a kreuzen	7122 TA	aalten
erik branten	7141 TS	groenlo
de watermolenschool	7141 VN	groenlo
w timmerman	7151 MK	eibergen
p h alma	7161 PH	neede
w van wanrooy	7221 AC	steenderen
mark hoefnagels	7321 DM	apeldoorn
j de vries	7322 EV	apeldoorn
r a h p rothengatter	7331 AJ	apeldoorn
g a van laere	7335 GR	apeldoorn
björn vlaskamp	7398 AC	terwolde
j w lubberhuizen	7451 NL	holten
b j kapper	7522 KM	enschede
j p dijkstra	7523 DP	enschede
j p e sunter	7523 SJ	enschede
hugo bosman	7531 LG	enschede
h joosten	7542 KZ	enschede
n p a braspenning	7542 VJ	enschede
t kers	7543 DH	enschede
n maassen	7543 VE	enschede

w van essen	7546 AD	enschede
j meenderink	7552 GT	hengelo ov
j g m oude nijhuis	7581 EZ	losser
j h stobbe	7608 GG	almelo
w e hagen	7642 TW	wierden
r varenhorst	7876 EC	valthermond
j meijer	8016 BB	zwolle
hans post	8022 AZ	zwolle
t l van heummen	8024 CN	zwolle
h j van de riet	8032 KT	zwolle
a f m sanders	8081 KB	elburg
l g e laproi	8096 BE	oldebroek
i bos	8101 ZC	raalte
h c j kroeders	8212 VH	lelystad
j poutsma	8261 GE	kampen
w g f jansens	8608 XJ	sneek
jan cor kars	8903 KA	leeuwarden
h b j ensing	9321 AS	peize
g meyer	9673 BT	winschoten
evert drijver	9742 LH	groningen
th j bijleveld	9756 BB	glimmen
w t g dresscher	9989 AB	warffum
j b ketel	9989 EW	warffum
w dietvorst	B-2000	antwerpen
a moreels	B-2710	antwerpen hoboken
marc fleurent	B-3281	averbode
g a van acker	B-8400	oostende
filip heyvaert	B-9391	baardegem-aalst
e j frenkel	CH-2009	neuchatel 9
r j maris	D-5210	troisdorf 15

ALFABETISCH INDEX OP DE LEDENLIJST

3314 AP aarnoudse, j
4901 AV aarts, m j
B-8400 acker, g a v
1098 HS albada, r va
7161 PH alma, p h
3911 AX arnold, w h
1054 LR baalen, j n v
1011 AZ bakker, k t
2481 XC bakker, r
1399 GK ballast, n h
4266 EL bant, jaap
5463 NW beerensteyn,
2402 BT bekkersschoo
1541 GV berkel, h va
5552 RB bierings, j g
9756 BB bijleveld, th
1082 LE blanken, r r
3448 XH bleker, j den
3607 GS boetekees, b
5012 GT bont, r j n m
8101 ZC bos, i
7531 LG bosman, hug
3742 AL boterenbrood
5017 HB bots, j j
7141 TS branten, erik
7542 VJ braspenning,
3828 RJ breukel, jan-
3811 HT bruggen, j va
1411 VW bruin, simon
4645 EZ bulthe, g
2317 CM burgt, j van
2624 PS davis, p b
1065 BH dekker, h g
2105 SM delta-t, tech
1521 TW denneman, g
1191 AK derksen, j w
2272 EL diepenbroek,
B-2000 dietvorst, w
2241 SB dijk, p w m va
7523 DP dijkstra, j p
1422 EK ditzel, j c l
4254 BT doesburg, e j
2101 HC doorduin, m w
6921 KT doorn, j p van
1055 JA drenthe
9989 AB dresscher, w t
2593 CW driessche, c v
9742 LH drijver, evert
2628 AM dubbelt, a
2724 PV dunnewold, b
3822 AD eekelen, j j va
4611 JX eekelen, j van
2264 DP eijkhoff, g p
1121 CP elfferich, a
9321 AS ensing, h b j
3076 CT ent, w r van d
7546 AD essen, w van
3271 XC etman, l
1324 MC fisser, g
B-3281 fleurent, marc
1191 HC floor, m a
1015 EA foeken, paul
CH-2009 frenkel, e j
1421 XK gerrits
3078 HD granville, j h
1007 EA groen, peter v
2596 AK haags montess
5550 AB haanstra, h b
2314 VS haarlem, j h c
7642 TW hagen, w e
1241 LV harmelen, m w
5501 HC hart, j j 't
3816 AW hauptmeijer, v
1211 KX hekman, a
2564 SR hendriks, g j
6983 ES hermans, j
2564 EL hessing, davy
5706 DM heummen, j va
8024 CN heummen, t l
B-9391 heyvaert, filip
2241 PC hiep, maurice
7321 DM hoefnagels, m
1094 LK hoek, j w van
1098 KL hoek, w a van
2722 BK hoeven, f van
6561 EX hoffman, d j
2105 XV hoogland, c
3181 RC horst, j ter
2285 HK huizenga, t
3077 BE hull, h m van
2274 VH iskandar, hasn
3742 XX jansen, r j
8608 XJ jansens, w g f
7542 KZ joosten, h
2317 LH jouvenaar, c w
4414 RS kampen, i van
3972 EG kamperman, t
7522 KM kapper, b j
8903 KA kars, jan cor
7543 DH kers, t
5345 DE kerstholt, m p
9989 EW ketel, j b
1094 KK klijn, c
1688 DA kok, a s p
2161 RV koning, j p
1784 AG koolstra, r h
2280 AD koswal, r
6661 BN kramer, p
6532 TN kranenburg, r
3603 GA kreuger, t
7122 TA kreuzen, a
2315 JN kroeders, davi
8212 VH kroeders, h c
6601 BD kruize, j a
6865 BH kuiken, j c j
1074 AD kulhan, a
3069 BK laar, a g m va

2551 EM ladenberg, a j
7335 GR laere, g a var
8096 BE laproi, l g e
2282 HL leger, m
5712 BR lemmen, g
1077 BB linden tol, w
2555 ZK linden, j g j v
4817 VB lottum, th va
7451 NL lubberhuizen,
4461 NH luijt, w m
7543 VE maassen, n
D-5210 maris, r j
7552 GT meenderink, j
8016 BB meijer, j
2624 LC meijer, m c
9673 HT meyer, g
1401 SR meyer, p
B-2710 moreels, a
1231 AP morgen, a vor
1214 AV mosca roman,
5612 PL naburgh, e p
1945 PT nijhuis, s a
2716 BN oosterink, m
1827 JA ooyevaar, a w
7581 EZ oude nijhuis, j
1827 NA paassen, n c j
2517 XT pallada, f w n
6325 BN peeters, a j m
4724 AJ perdaems, h a
5924 EA peters, j j
6531 RL pijnappel, l l
1782 AT ploeg, john va
3481 VP pol, h van der
8022 AZ post, hans
8261 GE poutsma, j
2316 HJ repe, c a van
8032 KT riet, h j van c
3512 XJ rijke, d de
6241 NK roijen, h e c
3831 AX roo, f de
7331 AJ rothengatter,
8081 KB sanders, a f m
3705 ZL scholten, jacol
2465 BB schrama, t g
1079 WR schuilenburg,
2715 VN silooy, e r
1013 NJ sluis, l
1324 CR spanningveld,
2343 DV speekman, w
4184 EV stappershoef,
2271 RA staring, p
5914 VR steinmann, h
3065 HA steketee, ger
1381 AJ steunebrink, l
1097 AS stevens, meni
1094 CZ stikker, n
7608 GG stobbe, j h
3902 GJ stolman, h
3311 CS straten, r vd
1061 CH strijdhaftig, c
7523 SJ sunter, j p e
3201 GH tak, willem
1033 NM tammer, f
7151 MK timmerman,
2162 HK touw, koos va
6001 BS tunissen, pete
1012 MK tusveld, mw l
1353 CH uijldert, t
2273 HS valk, j a c g
7876 EC varenhorst, r
4475 AG velde, n d va
5051 BT velden, m f j
2263 VX veldman, h g
5632 LA velthuys, h
2352 JM verbaken, a j
1218 CN verhaart, mw
7396 AC vlaskamp, bj²
1057 PG vlies, c van d
1403 GA vreedenburg,
7322 EV vries, j de
6535 TT vroegindeweij
6501 BC waart, a a de
2201 WG wal, sipke de
7221 AC wanrooy, w v
7141 VN watermolensc
1782 GJ weel, c m
1214 CP wielemaker, c
3705 ZE wijgerde, kees
2153 CE wijk, r s van d
3571 XD wiselius, g j w
1788 NX wit, f c a de
1121 CP woude, h h l v
3067 EZ wubben, j c
2811 AC zonneveld, leo
6706 GD zuidema, b
2036 CM zwaag, b van c
2811 AN zwaan, a vd
2584 RB zwan, n vd
1102 AA zwartjes, a f
2201 VK zwarts, j m s

inhoud on-line 17

NewBrain-
gebruikersgroep
postbus 4494
1009 AL amsterdam

softwarebibliotheek

BESTELFORMULIER

naam

adres

postcode en woonplaats

telefoon

hcc-lidmaatschapsnummer

bestelt bij deze de aan ommezijde aangegeven software

formaat aankruisen:	
☐ cassette	___ bestelnummers à ƒ 5,00 = ƒ ___
☐ diskette 200k	
☐ diskette 400k ss	
☐ diskette 400k ds	verzendkosten ƒ 2,50 ƒ ___
☐ diskette 800k	
☐ ___	TOTAAL ƒ ___

bovenstaand totaalbedrag

☐ is overgemaakt d d ___

op postrek 2505800 tnv hcc newbrain-gebruikersgroep amsterdam

afkomstig van bank-/postrek nr ___

tnv ___ te ___

☐ wordt voldaan met bijgesloten girobetaalkaart, betaal- of eurocheque

nummer giropas of betaalpas ___

handtekening

NewBrain-
gebruikersgroep
postbus 4494
1009 AL amsterdam

softwarebibliotheek

cassettesoftware
- ___ ASM-1
- ___ BWK-1
- ___ DIVERS-1
- ___ DIVERS-2
- ___ DIVERS-3
- ___ DIVERS-4
- ___ DIVERS-5
- ___ DIVERS-6
- ___ DIVERS-7
- ___ EDUC-1
- ___ EDUC-3
- ___ EDUC-4
- ___ FIN-1
- ___ GRAFI-1
- ___ GRAFI-2
- ___ HULP-1
- ___ HULP-2
- ___ HULP-3
- ___ MATH-1
- ___ MATH-2
- ___ MATH-3
- ___ MATH-4
- ___ MATH-5
- ___ NBFILES
- ___ ONLINE-5
- ___ ONLINE-6
- ___ ONLINE-7
- ___ ONLINE-8
- ___ ONLINE-9
- ___ ONLINE-10
- ___ SPEL-1
- ___ SPEL-2
- ___ SPEL-3
- ___ SPEL-4
- ___ SPEL-5
- ___ SPEL-6
- ___ SPEL-7
- ___ SPEL-8
- ___ SPEL-9
- ___ UGV-1
- ___ _____

*) = ook verkrijgbaar op cassette; de andere van de serie N... en de serie SCHIJF- alleen op diskette

- ___ N201
- ___ N202
- ___ N203
- ___ N204
- ___ N205
- ___ N206
- ___ N207
- ___ N208
- ___ N209
- ___ N210
- ___ N211
- ___ N212
- ___ N213
- ___ N214
- ___ N215
- ___ N216
- ___ N217
- ___ N218
- ___ N219
- ___ N220
- ___ N221
- ___ N222
- ___ N223
- ___ N224
- ___ N225
- ___ N226
- ___ N227
- ___ N228
- ___ N229
- ___ N230
- ___ N231
- ___ N232
- ___ N233
- ___ N234
- ___ N235
- ___ N307
- ___ N308
- ___ N309
- ___ N310
- ___ N311
- ___ N312
- ___ N313
- ___ N314
- ___ N315
- ___ N316
- ___ N401
- ___ N402
- ___ N501 *)
- ___ N502 *)
- ___ N503 *)
- ___ N504 *)
- ___ N505 *)
- ___ N506
- ___ N507
- ___ N508
- ___ N509 *)
- ___ N510 *)
- ___ N511 *)
- ___ N512
- ___ N513

- ___ SCHIJF-1*)
- ___ SCHIJF-2*)
- ___ SCHIJF-3*)
- ___ SCHIJF-4*)
- ___ SCHIJF-5*)
- ___ SCHIJF-6*)
- ___ SCHIJF-7*)
- ___ SCHIJF-8*)
- ___ SCHIJF-9*)
- ___ SCHIJF-10

per 10 delen in één bestelling het 11e gratis

volumes uit de programmatheek van de dos-gebruikersgroep
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ _____
- ___ catalogus-diskette

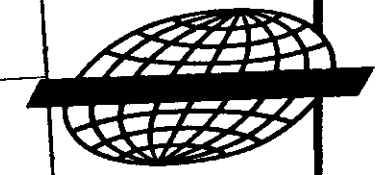
PRIJZEN
alle software, voor elk formaat, kost ƒ 5,00 per bestelnummer

plus bij bestelling per post ƒ 2,50 verzendkosten

buitenwereldkaart

De Buitenwereldkaart kan gebruikt worden in alle mogelijke NewBrain-systeemconfiguraties en bevat de volgende basiscomponenten:

PIO parallelle I/O-poort met 16 lijnen
CTC counter/timer chip
SIO dubbele seriële-poortchip (RS-232)



De buitenwereldpoorten:

- 2 x V24-poort volgens NewBrain-connectoraansluiting
- 1 x modem-/diversenpoort met enkele CTC-aansluitingen en DTR, DCD
- 1 x PIO-poort met alle PIO-poortaansluitingen
- 1 x expansiepoort: gebufferde databus met enkele I/O-selectlijnen
- 2 x voedingsaansluiting: voor doorlussen NewBrain-voeding

Toepassingen zijn onder andere:
- modemcommunicatie zonder flikkerende beelden (o. a. Fido)
- diverse RS-232-communicatie
- A/D-omzetting (oscilloscoop)
- D/A-omzetting (analoge besturing)
- pulsteller (lopende band)
- Centronics-printeraansturing
- modelbaanbesturing
- procescontrole

nu verkrijgbaar als aanvulling op de buitenwereldkaart

i/o-kaart

met een verscheidenheid aan mogelijkheden:

- 8 digitale uitgangen (via PIO aansturen)
- 8 digitale ingangen (via SEL 1 uitlezen)
- 2 min-aansluitpunten ten behoeve van de digitale ingangen
- 2 +12 V-aansluitingen en 4 +5 V-aansluitingen voor de digitale ingangen
- 2 aansluitingen voor de analoge ingang

geheel gemonteerd, zó in te pluggen, voor slechts ƒ 150,-

NewBrain-
gebruikersgroep
postbus 4494
1009 AL amsterdam

de tweede oplage is beschikbaar!
geheel gemonteerd, zó in te pluggen, voor maar ƒ 150,-
(inclusief software en een handleiding van 50 pagina's)
neem contact op met maarten floor, telefoon (02963) 4374