Burgerinformatica, een oriëntatie

B. van Asch/G.O. Visser
Ler. Opl. Z.W. Nederland, Delft

Samenvatting

De ervaringen van een nascholingscursus 'burgerinformatica' voor docenten van scholen die niet tot het zogenaamde '100-scholen-project' behoren, staan centraal in dit artikel.
Een nogal van het gangbare patroon afwijkende cursus, resulterend in verdeelde meningen achteraf.

Inleiding

In oktober 1984 kregen wij opdracht een nascholingscursus 'Oriëntatie Burgerinformatica', bestemd voor Rijksscholen te organiseren. Deze cursus was bestemd voor die Rijksscholen die niet aan het 100-scholen-project hadden deelgenomen, maar die van de minister toch toezegging tot nascholing op het gebied van de informatica hadden gekregen.
Uitgangspunt zou zijn dat het een cursus zou worden waarin men enerzijds enige vaardigheid zou ontwikkelen in het omgaan met een computer en anderzijds zouden de onderwerpen uit het 100-scholen-project (de zgn. modulen tekstverwerking, gegevens/bestanden etc.) ook min of meer aan bod komen. De cursus zou in totaal twee semesters beslaan. Het eerste deel heeft plaatsgevonden in de periode januari-juni 1985.

Wij hebben een paar redenen om iets over deze cursus te schrijven. In de eerste plaats week deze cursus op veel plaatsen nogal sterk af van veel andere (na)scholingscursussen die er op dit gebied zijn. Dit werd in niet geringe mate veroorzaakt door het feit dat het hier om vier verschillende scholen met vier verschillende typen computers ging. En in de tweede plaats zijn we bij de stukken over programmeren in Basic (min of meer machine-onafhankelijk) te werk gegaan volgens ideeën waarvan wij het waard vinden die te vermelden. Naast programmeren in Basic komen in dit artikel aan bod de start van de cursus en een beschrijving van een reeks activiteiten rond het thema tekstverwerking.

Het begin

Wij gingen ervan uit dat onze cursisten zich voor wat betreft computers op niveau nul bevonden.
Voorts hebben we ervoor gekozen de mensen zo snel mogelijk vertrouwd te maken met hun eigen type computer. Dit had kunnen gebeuren via een zeer gerichte reeks instructies; wij gaven er echter de voorkeur aan het op een 'opener' manier te doen. De cursisten kregen opdrachtbladen waar ze door onderzoeken en/of uitproberen open plekken moesten invullen.
Zoals bijv.:
- schakel de computer in d.m.v. .....
- op het scherm verschijnt .....
- veranderingen in getypte tekst kun je als volgt aanbrengen: .....
- plaats de schijf in het schijvenkastje; dit gaat als volgt: .....
- laat de computer het programma BABBEL vanaf de schijf gaan invoeren; daartoe moet je intypen .....

Van alle typen computers waren ter raadpleging handboeken aanwezig en door bovendien met twee- of drietallen aan één apparaat te werken kon men elkaar ook wel vaak verder helpen.
Wij hadden twee redenen om op deze manier te werk te gaan. De eerste was een heel praktische: deze werkwijze bespaarde ons de moeite om vier verschillende totaal uitgewerkte kennismakingspractica klaar te hebben. In plaats daarvan hadden we nu een reeks

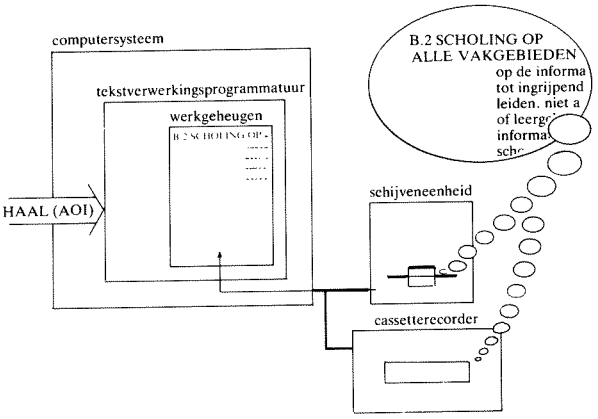
zeer algemene opdrachten, die bovendien ook nog op eventuele weer andere typen computers toepasbaar zijn. En in de tweede plaats hoopten we dat de cursisten op grond van hun eigen ervaringen hiermee zich een beeld zouden vormen op welke wijze de computer bij de eigen leerlingen te introduceren.
Met name ook toegespitst op de vraag hoe "open" je een introductie houdt, hoeveel je de leerlingen zelf uit laat zoeken. In het algemeen was men hierin toch terughoudend: "je moet leerlingen heel precies vertellen wat ze moeten doen, anders komt er niets van terecht."

Tekstverwerking

Na de introductie stond de eerste reeks bijeenkomsten in het teken van het thema 'tekstverwerking'. Dit was ook de eerste module uit het 100-scholen-project, en speciaal bij het kijken naar het "hoe" van tekstverwerking hebben we gebruik gemaakt van materiaal uit dit project. Als instap in het thema is genomen een standaardbrief van het postorderbedrijf Wehkamp, een zgn. persoonlijke brief. Vrijwel iedereen heeft hier regelmatig mee te maken: een fraai ogende brief in een nette lay-out die de suggestie moet wekken voor de geadresseerde persoonlijk getypt te zijn. Aan de hand van deze brief hebben we met name stil gestaan bij het begrip variabele: waarin verschilt de brief die aan de één wordt gezonden in de brief die aan een ander wordt gezonden; wat voor consequenties kan dat voor de tekst hebben. Om dit variabelenbegrip wat verder uit te diepen en tevens enkele nieuwe programmeer-opdrachten te introduceren hebben we het volgende eenvoudige standaardbriefprogrammaatje gebruikt:

```
10  PRINT "Dit wordt een brief waarin leden van
          een vereniging gemaand worden
          achterstallige contributie te betalen."
20  PRINT "Wat is de achternaam van het des-
          betreffende lid?"
30  INPUT A$
40  PRINT "Welke datum moet er op de brief?"
50  INPUT B$
60  PRINT "Over hoeveel maanden moet nog
          betaald worden?"
70  INPUT M
90  PRINT "Delft, ",B$
100 PRINT: PRINT: PRINT
110 PRINT "Geachte Heer/Mevrouw";A$
120 PRINT
130 PRINT "Omdat u nu reeds"; M; "maanden
          geen contributie heeft betaald moeten
          wij onkosten in rekening gaan
          brengen."°
140 PRINT "Wij verzoeken u binnen 14 dagen";
          M*5+3;" gulden aan ons over te
          maken."
150 PRINT: PRINT
160 PRINT "Hoogachtend,"
170 PRINT "het bestuur"
```

Uit een listing van dit programma komen een aantal aspecten naar boven. In de eerste plaats dat een programma een opeenvolgende reeks opdrachten is, waarbij in de taal Basic die hier gebruikt wordt elke regel van een nummer voorzien moet zijn. In de tweede plaats zijn in dit programma duidelijk variabelen te onderscheiden en ook is het duidelijk dat er verschillende soorten variabelen zijn. De ene soort kan als waarde alleen een getal aannemen, de andere soort kan als waarde woorden al dan niet in combinatie met andere symbolen en/of getallen aannemen. En tenslotte wordt in dit programma een Basic-opdracht geïntroduceerd om aan een variabele een waarde toe te kennen: de INPUT-opdracht. Het toekennen van een waarde geschiedt in dit geval via het toetsenbord. We hebben ook de LET-opdracht op dit moment geïntroduceerd, zijnde een manier om binnen het programma, zonder inmenging van buitenaf, aan variabelen waarden toe te kennen.
Een aspect van tekstverwerking komt in dit programma al naar voren: namen kunnen moeiteloos ingevoerd worden en er rolt een persoonlijk geschreven brief uit het apparaat. Toch is ook duidelijk dat het slechts een zeer gebrekkige vorm van tekstverwerking is, bv. de lay-out zal mogelijk niet fraai zijn, daarin voorziet het programma nu eenmaal niet.
Na de instap in het thema tekstverwerking (en in het programmeren in Basic) is eerst het "hoe" van tekstverwerking aan de orde geweest en vervolgens het praktisch werken met tekstverwerkingsprogramma's. Bij het "hoe" is veelvuldig gebruik gemaakt van teksten en schematische tekeningen uit module 1 uit het 100-scholen-project.
Een typerend plaatje:



Dit geeft niet alleen een overzicht van de verschillende onderdelen waaruit een als tekstverwerker fungerend computersysteem bestaat (er zit trouwens een fout in deze tekening, waar?), maar geeft ook de mogelijkheid om het effect van de verschillende commando's op verschillende plaatsen in het systeem duidelijk te maken.
De commando's waren in deze algemene (d.w.z. niet aan één specifiek tekstverwerkingsprogramma gebonden) setting geformuleerd als:

HAAL (gevolgd door de naam van een tekst)
VENSTER NEER, OP, LINKS, RECHTS
RECHTS, LINKS, OP, NEER (cursorbewegingen)
WIS KARAKTER
VOEG IN KARAKTER
ZOEK
VERVANG
DRUK AF
SLA OP

Het formuleren van deze algemene commando's was speciaal in onze situatie met betrekking tot die verschillende apparaten, dus programma's, erg handig. Door met deze reeks commando's een aantal 'droogzwem'-oefeningen op een stuk tekst uit te voeren wordt enig inzicht verkregen in wat er feitelijk gebeurt bij tekstverwerking. Met behulp van een overheadprojector zijn de resultaten van VENSTER OP, etc. op een stuk tekst prima zichtbaar te maken. Het beeld dat gebruikt werd (nl. het scherm is een soort venster waardoor je op een stuk van de tekst kunt kijken) bleek heel goed te voldoen. De volgende stap was het vertalen van de algemene reeks commando's naar de specifieke tekstverwerkingsprogramma's van de diverse apparaten. Vrijwel automatisch ontstonden er vergelijkingen tussen de verschillende apparaten en programma's: waar bij de één slechts aanraken van één toets nodig was vergde hetzelfde commando bij een andere een hele reeks toetsaanslagen.

Bij het werken met de tekstverwerkingsprogramma's stonden ons twee doelen voor ogen:
- enige vaardigheid opdoen met dat programma;
- zelf het gehele proces van tekstverwerking (van idee voor een tekst tot uiteindelijk produkt) doorlopen.

Het eerste is nagestreefd door een reeks manipulaties met een reeds op schijf staande tekst te laten uitvoeren.
Opdrachten daarbij waren o.a.:
- zoek uit hoeveel regels de tekst heeft;
- vervang bepaalde woorden door andere;
- verplaats stukken tekst;
- herschrijf het verhaal van een hij-verhaal naar een ik-verhaal.

Het tweede deel is nagestreefd door alle cursisten zelf een stuk tekst te laten ontwerpen aan de hand van door ons gestelde vragen over maatschappelijke consequenties van de intrede van tekstverwerkingsapparatuur. Deze vragen gingen o.a. over gevolgen voor de werkgelegenheid, maar ook over gevolgen voor het vak Nederlands. Nadat dit stuk tekst gemaakt was werd het op schijf opgeslagen. Vervolgens kreeg men een aantal artikelen over deze maatschappelijke aspecten mee naar huis. En in de daaropvolgende bijeenkomst was de eerste activiteit de oude tekst die ze zelf gemaakt hadden weer terughalen en zo nodig (mogelijk onder invloed van het gelezene) te veranderen. Toen men tenslotte over de eigen teksten tevreden was konden deze afgedrukt worden. Op deze manier had men in kort bestek het hele proces 'tekstverwerking' doorlopen, uitgaande van de volgende 'definitie'; tekstverwerking is het proces dat alle handelingen en bewerkingen omvat die nodig zijn om van een gedachte te komen tot een geschreven, getypt of gedrukt document. Ter afsluiting van het blok tekstverwerking is uitvoerig aandacht besteed aan mogelijke toepassingen in het onderwijs.
Vragen die je daarbij moet beantwoorden:
- moet de nadruk liggen op het vertrouwd raken met tekstverwerkingsapparatuur of meer op maatschappelijke aspecten;
- moeten leerlingen die met een tekstverwerker gaan werken typevaardigheid hebben, of is dit juist een middel om ze te leren typen;
- moet dit onderdeel in een apart vak burgerinformatica gestopt worden, of moet het specifiek op een aantal vakken (b.v. Nederlands) gericht worden;
- hoe evalueer je een en ander met je leerlingen.

Dat vraagt om...

Omdat aan de cursus burgerinformatica voor de Rijksscholen geen aparte gebruikerscursus was vooraf gegaan zoals bij de 100 scholen waren we gehouden ook wat aandacht aan programmeren te geven.
Bij het inpassen van programmeren in een burgerinformatica-cursus hebben wij ons door de volgende gedachten laten leiden:
a. Goede software moet je kopen en niet zelf maken.
b. Het net zo goed leren programmeren als sommige van je leerlingen kun je vergeten.
c. Zelf een programmaatje maken geeft inzicht in de werking van de computer.
d. Zelf een klein programma maken geeft inzicht in de werking van de gekochte software.
e. Zelf een programmaatje maken geeft gevoel van macht over, zelfvertrouwen t.o.v. de computer.
f. We nemen als programmeertaal de taal BASIC.

We zullen proberen aan de hand van wat voorbeelden uit de cursus te laten zien hoe we die gedachten concreet hebben gemaakt.
Maar eerst nog even een korte verantwoording van de punten a t/m e.
a. Bij het zelf maken van programma's bleek het ons ook steeds weer dat ze leuk zijn als demonstratie. Maar software die je echt door je leerlingen wilt laten gebruiken moet aan zoveel eisen voldoen dat dat alleen al qua tijd ondoenlijk is om daar naar te streven.
Bovendien hebben we een aantal keren behoorlijk onze neus gestoten toen wij programma's wilden gebruiken die door (goedwillende) amateurs gemaakt waren.
Daar komt nog bij dat het zelf maken van software het doel van de cursus behoorlijk voorbij zou streven.
b. In vorige cursussen was ons al vaker gebleken dat sommige mensen pas burgerinformatica willen geven als ze net zo handig of handiger zijn in programmeren dan hun beste leerlingen. Dit is op zich niet zo'n vreemde gedachte voor onderwijsmensen die gewend zijn zelf ver boven de stof te staan.

Voor een vak als burgerinformatica is dat echter onhaalbaar omdat:
1. we daar niet een hele studie in volgen;
2. de veranderingen zich zo snel voltrekken dat we wel nooit aan onderwijzen van het vak toe zullen komen omdat we onszelf moeten blijven scholen.

Alleen al het vaststellen van die uitgangsgedachte zou een hoop mensen een last van de schouders kunnen nemen almaar slimmer te moeten worden. Naar ons idee ontbreekt die gedachte nog te zeer in vele informaticacursussen. Men bedenke dat die leerlingen die zo handig zijn met programmeren meestal op de vingers van één hand te tellen zijn.

En wij denken dat het meer de taak is van onderwijsmensen om de grote achterblijvende massa iets bij te brengen over computers in plaats van jezelf als leerkracht voortdurend te meten met die paar handigerds.

c. Na eerst het programmeren te hebben gerelativeerd dan nu toch een aantal argumenten vóór.
Het spreekt voor zich dat het zelf sleutelen aan een programma je dwingt te bedenken hoe de computer dit uitwerkt.

d. In het vak burgerinformatica maak je met toepassingsprogrammatuur kennis. Eén van de doelen is om meer inzicht in de werking van zulke programma's te krijgen. Het zelf maken van een afspiegeling van zo'n programma vergroot dat inzicht.

e. Om een programma draaiend te krijgen moet je de computer de opdrachten precies in de goede volgorde en tot in alle details uitgewerkt voorleggen. Eenvoudige vergissingen zoals spelfouten en/of het weglaten van leestekens worden door een machine niet begrepen. Een en ander relativeert de macht van de computer.

f. Over de taal BASIC wordt altijd zeer negatief geschreven. Toch waren er een aantal argumenten die ons voor die taal deden kiezen.
   - Op ieder van de apparaten kon ermee gewerkt worden.
   - De taal is zeer gemakkelijk toegankelijk, verschillende vakrichtingen bleken geen probleem.
   - Echte grote programma's waarin een strakke structuur een must is maken we toch niet in deze cursus.

In de vorige paragraaf hebben we al beschreven hoe we via de opdrachten PRINT LET INPUT met een standaardbriefprogramma de opstap hebben gemaakt naar het toepassingsprogramma voor de tekstverwerker. Voor ons was dat een manier om de uitgangspunten c. en d. te verwerken in de cursus.

Kleine programmaatjes kon men nu maken maar er was al snel behoefte aan meer BASIC-opdrachten. Want wat doe je bijvoorbeeld als je in een standaardbriefje, afhankelijk van het antwoord op de vraag of de leerling een meisje of een jongen is, de brief wilt laten aanpassen? Dat kan via allemaal input-opdrachten, dat kan ook door de computer wat meer keuzewerk voor je te laten doen. *Dit vraagt om* de IF ... THEN ... ELSE opdracht.

Met onderstaand programmaatje hebben wij dit statement geïntroduceerd. Hierbij dus weer een beetje voortbordurend op het grotere toepassingsprogramma.

```
10  PRINT "Dit wordt een brief waarin ouders
        gewaarschuwd worden"
20  INPUT "Is de brief gericht aan de vader of de
        moeder?"; G$
30  INPUT "Wat is de achternaam"; A$
40  INPUT "Is de leerling een jongen of een
        meisje?" L$
50  INPUT "Wat is de voornaam van de leerling,";
        V$

70  IF G$ = "vader" THEN PRINT "Geachte
        Heer" ; A$
80  IF G$ = "moeder" THEN PRINT "Geachte
        Mevrouw"; A$
90  PRINT: PRINT
100 IF L$ = "jongen" THEN C$ = "zoon"
110 IF L$ = "jongen" THEN D$ = "hem"
120 IF L$ = "meisje" THEN C$ = "dochter"
130 IF L$ = "meisje" THEN D$ = "haar"
140 PRINT "Tijdens de lessen informatica is het
        ons opgevallen"
150 PRINT "dat uw"; C$;" ";V$;" verschijnselen
        van verslaving"
160 PRINT "aan de computer begint te vertonen"
170 PRINT "Het lijkt ons dan ook het beste dat";
        D$ "voorlopig"
180 PRINT "elk contact met een computer wordt
        ontzegd"
190 PRINT: PRINT
200 PRINT "          "; "Hoogachtend,"
210 PRINT "          "; "De Directie"
```

Een aardige en niet te moeilijke programmeeropgave hierbij was:

> Maak een programma "kwis" genaamd. In dit programma moeten een aantal meerkeuzevragen verwerkt worden. De gebruiker van dit programma moet deze vragen beantwoorden en aan het eind van het programma verschijnt de behaalde score.

Bij het maken van zo'n programma bleek steeds weer dat het leuk is voor demonstratie maar niet voor serieus gebruik. Want de computer moet precies dat ene antwoord hebben en verklaart alle mogelijke andere manieren om bijvoorbeeld het antwoord één òf 1 te geven als fout.
Daartegen zijn programma's natuurlijk te beveiligen, maar dan denken wij toch dat je goede software beter kunt kopen.

Het invoeren van de FOR ... NEXT-opdracht deden we aan de hand van de IF ... THEN ... ELSE opdracht.

vb.
```
10 PRINT "Moet dit zo nog langer doorgaan."
20 T = T + 1
30 T < 15 THEN 10
40 PRINT "Goed dit was het dan".
```

De opdracht uit regel 10 kun je eindeloos herhalen met GOTO (foei), je kunt het ook beperken tot 15 keer door met IF ... THEN te werken. *Dit vraagt* om een herhalingsopdracht zoals FOR ... NEXT.
We lieten een aantal kleine voorbeelden zien waarin:
a. onder- en bovengrens wisselend waren;
b. de initiële variabele I wel òf niet in de opdracht was verwerkt.

Er bleek toch steeds behoefte te bestaan aan veel kleine voorbeelden waarin dan weer even werd nagegaan wat die programmaatjes precies deden.

Door het stroomschema van FOR ... NEXT en IF ... THEN met elkaar te vergelijken ontdekten we:
1. de overeenkomst tussen die twee;
2. dat de opdrachten tussen FOR ... NEXT minstens altijd één keer worden doorlopen en afhankelijk van de vraag of de lopende variabele de bovengrens al is gepasseerd;

3. dat na afloop van een FOR ... NEXT-lus de lopende variabele hoger (of lager) is dan de grens tot waar hij mag komen.

Die laatste twee punten zijn altijd handig om te weten wanneer je niet ziet waar de fout in het programma zit (was ook onze ervaring). We besteedden ook nog aandacht aan dubbele lussen. Het maken van een digitale klok werd hierbij als uitdagende opdracht ervaren. Verder is het erg moeilijk zinvolle programmeeropdrachten te vinden waarin een dubbele lus voorkomt.

Wanneer we bij het stellen van meerkeuze vragen steeds dezelfde vragen gebruiken alleen over een ander onderwerp dan moet dat korter kunnen. Uitgaande van de vroegere kwisopdracht schotelden we ze eerst het volgende programma voor:

```
 10 CLS
 20 PRINT "WAT IS DE HOOFDSTAD VAN
    BELGIE?"
 30 INPUT A$
 40 IF A$ < > "BRUSSEL" THEN PRINT
    "FOUT"
 50 FOR I = 1 TO 1000: NEXT I
 60 CLS
 70 PRINT "WAT IS DE HOOFDSTAD VAN
    DENEMARKEN?"
 80 INPUT A$
 90 IF A$ < > "KOPENHAGEN" THEN PRINT
    "FOUT"
100 FOR I = 1 TO 1000: NEXT I
110 CLS
120 PRINT "WAT IS DE HOOFDSTAD VAN
    SPANJE?"
130 INPUT A$
140 IF A$ < > "MADRID" THEN PRINT
    "FOUT"
150 FOR I = 1 TO 1000: NEXT I
160 CLS
170 PRINT "WAT IS DE HOOFDSTAD VAN
    OOSTENRIJK?"
180 INPUT A$
190 IF A$ < > "WENEN" THEN PRINT "FOUT"
200 FOR I = 1 TO 1000: NEXT I
210 CLS
220 PRINT "WAT IS DE HOOFDSTAD VAN
    NOORWEGEN?"
230 INPUT A$
240 IF A$ < > "OSLO" THEN PRINT "FOUT"
```

Dit vraagt om een mogelijkheid die wisselende gegevens ergens op te slaan en in te lezen.

```
 20 FOR K = 1 TO 5
 30 CLS
 40 READ L$, H$
 50 PRINT "WAT IS DE HOOFDSTAD VAN";
    L$ "?"
 60 INPUT A$
 70 IF H$ < > A$ THEN PRINT "FOUT"
 80 FOR I = 1 TO 1000: NEXT I
 90 NEXT K
100 DATA "BELGIE", "BRUSSEL", "DENE-
    MARKEN", "KOPENHAGEN", "SPANJE",
    "MADRID", "OOSTENRIJK", "WENEN",
    "NOORWEGEN", "OSLO".
```

Het verschil in lengte van de twee programma's is heerlijk overtuigend.

Met de tot op dit moment opgedane programmeerervaring moest iets groots toch tot de mogelijkheden behoren.
We kozen voor het volgende:

Opdracht
Het wereldkampioenschap hardrijden op de schaats in Hamar nadert z'n voltooiing. Veertien van de in totaal zestien rijders hebben de laatste afstand (10 km.) reeds achter de rug, er rest nog één rit. Na de eerste zeven ritten op de 10 km ziet de voorlopige ranglijst met de eindtotalen er als volgt uit:

1.	Vergeer	166.931
2.	Bozjew	167.679
3.	Van der Duim	167.718
4.	Schalij	168.323
5.	Hadschieff	168.378
6.	Silk	168.466
7.	Falk-Larssen	168.855
8.	Eminger	169.214
9.	Hopman	170.166
10.	Oxholm	170.180
11.	Jäger	171.094
12.	Nilsen	172.347
13.	Baltes	173.225
14.	Sjasjerin	179.378

De laatste rit gaat tussen de Noor Karlstad en de Fin Niitylae. De resultaten van deze beide rijders op de eerste drie afstanden (500, 5000 en 1500 meter) zijn resp.
40.67 – 7.10.30 – 2.02.14
39.96 – 7.19.09 – 2.02.13

Zoals (waarschijnlijk) bekend levert elk resultaat op een afstand een bepaald aantal punten op. Deze punten worden berekend door voor elke afstand de score te herleiden tot de score per 500 meter. Voor de 5000 meter betekent dit b.v. dat de tijd eerst wordt omgezet in seconden en vervolgens wordt dat getal door 10 gedeeld. Deze uitkomst levert het aantal punten voor de 5000 m.

Maak een programma dat achtereenvolgens:
- op het scherm het tussenklassement na 7 ritten op de 10 km vertoont;
- vraagt om de invoer van de gegevens van de laatste rit;
- de puntentotalen van de beide laatste rijders berekent;
- het eindklassement van alle 16 rijders met hun puntentotalen vertoont.

De laatste rit werd gewonnen door Karlstad in de tijd van 14.48.54; Niitylae scoorde 15.13.73.

Na dit te hebben aangeboden en de meesten zich wat wanhopig afvroegen wat nu, hebben we het volgende gedaan:
- uitgebreid de verschillende deelproblemen van het geheel besproken;
- ze gevraagd het werk te verdelen, ieder maakt een stukje.

Toen bleek het allemaal best mee te vallen en kwam er heel wat uit de bus.
Dit was nou typisch een probleem waar ieder wat mee kon. Wie het moeilijkste deel wil, neemt dat, wie dat niet wil, neemt wat anders. Een heterogene groep kan hiermee aan de slag.
De cursisten brachten hier ook zelf differentiatie in de opdracht aan zoals het op verschillende manieren tot op 3 cijfers achter de komma afbreken, de één hield wel rekening met afronding, de ander brak het gewoon af.
Natuurlijk kun je hier opmerkingen maken dat deze opdracht de werkelijke gang van zaken geweld aandoet.
Onze ervaring is toch dat men gemotiveerd raakte door het betrekkelijk realistische van het probleem.

Tot slot van het onderdeel programmeren hebben we een hele serie opdrachten aangeboden.
Deze vielen in drie categorieën uiteen:

I Programma's waar men wijzigingen in moest aanbrengen.
II Programma's waarvan men de strekking moest vaststellen en eventueel nog wat verklarende tekst kon toevoegen.
III Het maken van grotere programma's.

We hadden hierbij de bedoeling te laten zien dat een opdracht het programmeren betreffend niet altijd hoeft te behelzen: maak een heel programma. Dit laatste is iets waar leerlingen vaak tegenaan hikken.

Aan het eind van deze serie lessen constateren we dat:
- we hebben geprobeerd via een logische opbouw van de lesstof ieder van de volgende programmeeropdrachten als een vanzelfsprekendheid in te voeren;
- men niet echt grote programma's kan maken;
- men ideeën en opdrachten heeft aangedragen gekregen die ook in de eigen lessen gebruikt kunnen worden.

Slot

Door de hele cursus heen zijn we op verschillende manieren en op verschillende niveau's met de stof bezig geweest. Aan de ene kant zijn de cursisten op eigen niveau bezig geweest, met name in het ontwikkelen van enige vaardigheid in het omgaan met een computer en met het zich eigen maken van de eerste beginselen van programmeren. Aan de andere kant zijn er ook meerdere momenten geweest waarop opdrachten op leerling-niveau moesten worden uitgevoerd en waar we als onderwijsgevende gekeken hebben naar mogelijkheden om een en ander in de klas toe te passen.

Bij de nabespreking waren de meningen zeer verdeeld. Een aantal mensen was er zeer tevreden over en vast van plan veel van de materialen zonder meer te gaan gebruiken. Er was echter ook een groep die het werken op leerling-niveau in een dergelijke cursus niet zo zag zitten. Zij wilden liever stof op eigen niveau aangeboden krijgen, want "vertalen naar leerling-niveau en klassesituatie is iets dat wij dagelijks doen."

Hier deed zich een probleem voor dat we al eerder waren tegengekomen bij de nascholing. Namelijk de keuze tussen geschoold worden om zelf nog meer technische/zakelijke kennis op te doen òf geschoold worden om leerstof te kunnen bewerken/aanbieden geschikt voor jouw leerlingen. De keuze voor het eerste lijkt moeilijker want de stof is van een hoger niveau. Toch zijn wij van mening dat de keuze voor het laatste vaak nog wel zo moeilijk is en het één van de belangrijkste taken van een onderwijsgevende is om zich ook hierin te verdiepen. Daarom hebben we dat laatste niet laten liggen.