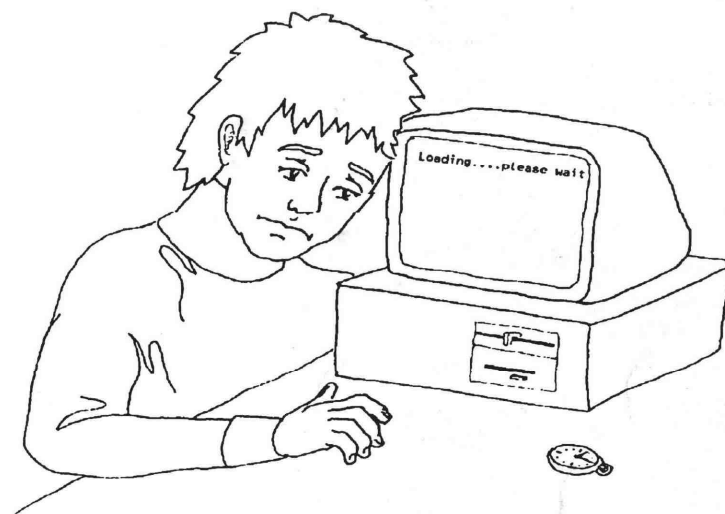




Negentiende jaargang nr. 1
Voorjaar 1995
88



In dit nummer o.a.:

De laatste μ P Kenner?
Cursus C: Nog een keer

Inhoudsopgave

De μ P Kenner

Nummer 88, voorjaar 1995
Verschijnt onregelmatig...
Oplage: 250 stuks
Druk: FEBO Offset, Enschede

De redactie:

Nico de Vries
Gert van Opbroek
Geert Stappers

Eindredactie:

Nico de Vries

Vormgeving:

Nico de Vries

Redactieadres:

p/a Nico de Vries
Van der Waalsstraat 46
2984 EP Ridderkerk

De μ P Kenner nummer 89 verschijnt
wanneer er voldoende kopij voor is...

Kopijsluitingsdatum voor nummer 89 is
dan ook onbepaald.

Vereniging

Uitnodiging voor de clubbijeenkomst	5
Algemene ledenvergadering van de KIM Gebruikersclub	
Nederland	6
Van de Voorzitter	21
Voortgang KGN68k (deel 18)	21

Algemeen

Redactioneel	4
--------------------	---

Software

C++ de opvolger van C (deel 2)	11
--------------------------------------	----

Hardware

Beweging	7
Beweging, deel 2	18

De μ P Kenner is het huisorgaan van de KIM gebruikersclub Nederland en wordt bij verschijnen gratis toegezonden aan alle leden van de club. De μ P Kenner verschijnt in principe vijf maal per jaar, telkens op de derde zaterdag van de maanden februari, april, augustus, oktober en december.

Kopij voor het blad dient bij voorkeur van de leden afkomstig te zijn. Deze kopij kan op papier, maar liever in machine-leesbare vorm opgestuurd worden aan het redactieadres. Kopij kan ook op het Bulletin Board van de vereniging gepost worden in de redactie area. Nadere informatie kan bij het redactieadres of via het bulletin board opgevraagd worden.

De redactie houdt zich het recht voor kopij zonder voorafgaand bericht niet of slechts gedeeltelijk te plaatsen of te wijzigen. Geplaatste artikelen blijven het eigendom van de auteur en mogen niet zonder diens voorafgaande schriftelijke toestemming door derden gepubliceerd worden, in welke vorm dan ook.

De redactie noch het bestuur kan verantwoordelijk gesteld worden voor toepassing(en) van de geplaatste kopij.

Redactioneel

Ook op de bijeenkomst in Krommenie geweest? Waarschijnlijk niet, want onze meest populaire lokatie was deze keer maar matig bezocht. Toch een leuke bijeenkomst, want er is een fraaie inleiding over multi-media gehouden door Co Filmer, en er waren toch weer leuke koopjes. Dat waren, naast het redelijke weer die dag, de positieve zaken. Want negatieve waren er ook.

Zoals u waarschijnlijk wel weet, behoort een bestuurslid democratisch aangewezen (pardon gekozen) te worden, en mag zo'n persoon vervolgens gedurende maximaal twee jaar het reilen en zeilen van de club gaan beïnvloeden. Die bestuurswisselingen treden normaal op gedurende de laatste bijeenkomst van het jaar, die in november. Toen hadden we een geweldige opkomst: zes bestuursleden en twee leden. En daarmee kun je geen ledenvergadering houden. Ergo: het bestuur bleef in eerste instantie zoals het was.

Hierop werd besloten de ledenvergadering te verplaatsen naar de bijeenkomst van januari. In Krommenie hebben we zo goed en zo kwaad dat ging dan ook een ledenvergadering proberen te houden, maar dat ging niet zo best: er was slechts één bestuurslid aanwezig! Ook een oproep voor nieuwe bestuursleden leverde slechts één vrijwilliger op. En daarmee hebben we club met een incompleet bestuur, want de penningmeester en de secretaris ontbreken. Ook de positie van redacteur voor dit geschrift is vacant, want deze woorden worden aan het magnetisme toevertrouwd door de layouter. De mensen die hiervoor deze functie hebben bekleed kun je geen ontrouw verwijten: zij hebben dat zonder uitzondering meer dan 6 jaar gedaan en zijn al veel langer actief in de club geweest. En ook zij willen wel eens wat anders.

Kijken we naar een andere graadmeter voor de ledenbelangstelling dan ziet het geheel er ook niet rooskleurig uit. Ik heb het over de aanbod van kopij. Effectief wordt het blad nog steeds volgepend door een drietal mensen. En dat is ter merken aan de dikte van het blad en inmiddels ook aan de verschijningsfrequentie. Inmiddels is door bestuur en redactie (of beter gezegd wat dat nog van over is) besloten het blad uit te brengen als er een aanleiding voor is, of als er voldoende kopij is.

Toen is aan het publiek gevraagd hoe nu verder te gaan, want effectief hebben we met z'n allen nog wel een vereniging, maar eenje zonder een bestuur. En uit de leden is kennelijk geen nieuw bestuur samen te stellen, want er zijn niet voldoende kandidaten. Om kort te gaan: het lijkt erop dat de vereniging niet langer door de leden gedragen wordt. Eén van de aanwezigen stelde toen, dat er serieus nagedacht moet worden over nieuwe wegen, waarvan de vereniging opheffen er ook een is.

**Om kort te gaan:
het lijkt erop dat de
vereniging niet
langer door de
leden gedragen
wordt.**

U mag dus raden wat er op de agenda staat van de bijeenkomst van 22 april aanstaande. Jawel: de toekomst van de club. U mag ook raden wat er gebeurt als die ledenvergadering net zo verloopt als in die Krommenie. Jawel: dan is dit de laatste uP Kenner.

U bepaalt of dat gebeurt of niet!

Nico de Vries

Uitnodiging voor de clubbijeenkomst

Datum: 22 april 1995
 Locatie: Wijkcentrum 't Veurbrook
 Jan Tooropstraat 27
 7606 Almelo
 Tel.: 05490 - 10353
 Thema: Toekomst van de vereniging

Routebeschrijving

Vanuit het westen en het zuiden (A1/A35):

1. Aan het einde van de snelweg rechtsaf. Bij het eerstvolgende kruispunt MET VERKEERSLICHTEN linksaf, richting Wierden/Zwolle. Bij de eerstvolgende verkeerslichten recht door. Bij de volgende verkeerslichten (links BP tankstation en Opel garage Kamp) gaat u rechtsaf.

2. U rijdt nu op de Windmolenbroeksweg. Doorrijden tot over de brug, dan de eerste straat rechts. Dit is de W. van Konijnenburgstraat. Na plm. 50 meter rechtsaf. Dit is de Tooropstraat. Met de bocht mee naar links. Na plm. 50 meter aan de rechterkant: 't Veurbrook.

Vanuit het noorden (via de N 36):

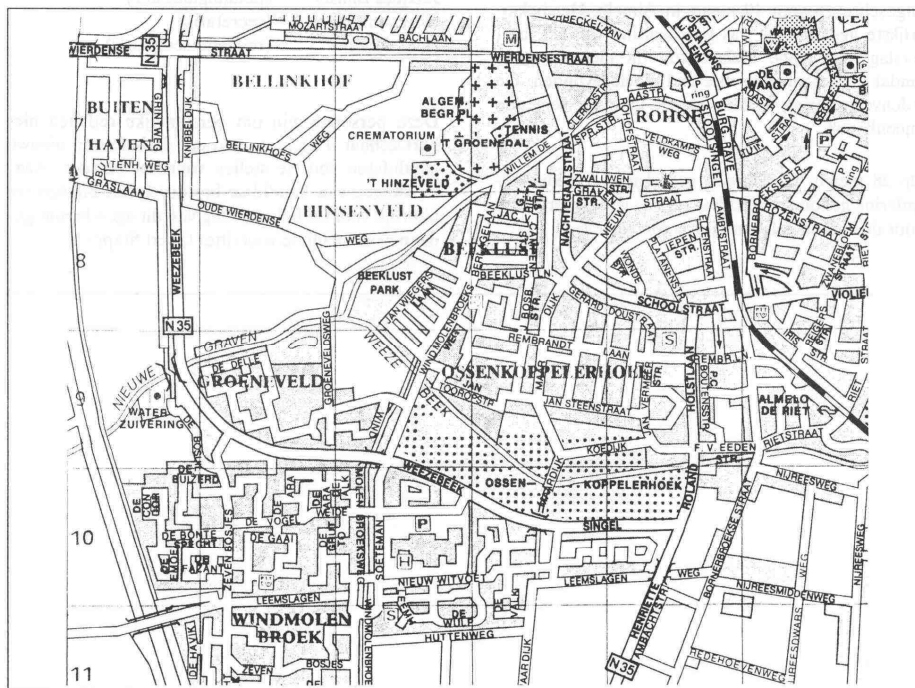
1. Bij de stoplichten rechtsaf, richting streekziekenhuis. U bevindt zich nu op de rondweg om Almelo. Deze weg blijven volgen tot u het BP tankstation ziet bij dit kruispunt linksaf. Zie verder punt 2.

Met openbaar vervoer:

Vanaf NS-station Almelo met de stadsbus naar de wijk Molenbroek. Uitstappen bij de halte Windmolenbroeksweg. Schuin tegenover de bushalte staat een wegwijzer, daarop staat ook 't Veurbrook vermeld.

Programma:

9:30	Zaal open met koffie
10:15	Opening
10:30	Ledenvergadering/discussie over hoe het verder moet met de vereniging. KOMT ALLEN!!!
11:30	Forum en Markt
12:00	Lunch, consumpties tegen betaling Aansluitend het informele gedeelte met de mogelijkheid om andermans systemen te bewonderen en Public Domain software uit te wisselen. U en uw systeem zijn uiteraard van harte welkom.
17:00	Sluiting.



Algemene ledenvergadering van de KIM Gebruikersclub Nederland

Omdat de ledenvergaderingen van november 1994 en januari 1995 effectief geen doorgang hebben gevonden wegens gebrek aan belangstelling, wordt opnieuw een ledenvergadering gehouden op 22 april.

Datum: 22 april

Plaats: Almelo

Voorgestelde agenda:

- 1: Opening, vaststelling agenda
- 2: Verslag van de vorige ledenvergadering
- 3: Mededelingen
Status project KGN68k
Status DOS-65
- 4: Concept-begroting 1994
- 5: Verkiezing kascontrole-commissie 1994
- 6: Toekomst van de vereniging: opheffen, doorgaan in kleiner en informeel verband, of net doen of onze neus bloedt?
- 7: Verkiezing bestuursleden
- 8: Eventuele overige agendapunten
- 9: Rondvraag

Toelichting:

Ad. 2:

De vorige ledenvergadering was in eerste instantie uitgeschreven voor 19 maart in Almelo. Het belangrijkste agendapunt was goedkeuring van het jaarverslag over 1994. Helaas kon dit niet doorgaan omdat de kascontrole niet had plaatsgevonden. De ledenvergadering is toen verzet naar de volgende bijeenkomst.

Op 28 mei in Haarlem is vervolgens de ledenvergadering gehouden. Het jaarverslag was ondertussen door de kascontrolecommissie goedgekeurd.

Behandelde agendapunten:

- 1: Het verslag van de ledenvergadering d.d. 25-09-1993 zoals afgedrukt in μ P Kenner 83 is goedgekeurd.
- 2: Het jaarverslag werd door de leden, na enige uitleg van de penningmeester en op voorstel van de commissie kascontrole, goedgekeurd.
- 3: Er waren geen overige agendapunten of punten voor de rondvraag.

Een kopie van het goedgekeurde jaarverslag is op te vragen bij de voormalige secretaris of de voormalige penningmeester.

Ad 6:

De vereniging verkeert thans in een crisis: de opkomst op bijeenkomsten is minimaal, de kopijstroom voor het clubblad is vrijwel opgedroogd, en er zijn drie vacatures in het bestuur die nog steeds niet ingevuld zijn. Er moet daarom worden nagedacht over de toekomst van de club, want kennelijk is er nauwelijks bestaansrecht voor.

Ad 7:

Vorig jaar zijn als bestuursleden afgetreden:

Jacques Banser	(penningmeester)
Gert van Opbroek	(secretaris)
Nico de Vries	(redactie-secretaris)

Deze personen zijn om persoonlijke redenen niet herkiesbaar. Het bestuur roept de leden op nieuwe kandidaten voor te stellen voor het bestuur. Aanmeldingen van kandidaat-bestuursleden kunnen tot uiterlijk voor de behandeling van dit agendapunt gedaan worden bij de voorzitter Geert Stappers.

Beweging

Binnen onze club zijn de meeste mensen bezig met software. Daarbij blijft alles wat er gebeurt altijd IN de computer. Al jaren terug vond ik het eigenlijk veel interessanter om met de computer iets te doen met de grote buitenwereld. Dus bijvoorbeeld iets laten bewegen of besturen.

Alle begin is moeilijk, dus de eerste experimenten bleven heel eenvoudig en verder kwam ik toen eigenlijk ook niet. Tot 2 jaar geleden een familielid eens informeerde of het mogelijk is om zijn draaibank met de computer te besturen. Het antwoord is: Ja natuurlijk, kijk maar naar alle CNC draaimachines die er overal in de fabrieken te vinden zijn. Toen we er echt aan begonnen kwamen de problemen pas. Omdat dit een hobbymachine is, hebben we gebruik gemaakt van stappenmotoren. Inmiddels hebben we een werkende machine.

Op de bijeenkomst in april zal ik e.e.a. vertellen over dit project en uiteraard komt daarbij ook een demonstratie.

Als inleiding zou ik hier natuurlijk allerlei details kunnen vertellen maar dan zou men de onjuiste indruk kunnen krijgen dat stappenmotoren erg inge-

wikkeld zijn. Omdat we toch eigenlijk een club zijn van zelf-doen en zelf bouwen, lijkt het me leuker om te beginnen met een beschrijving van een zeer eenvoudige stappenmotor besturing.

Dit is zo eenvoudig, het mag helemaal geen besturing heten, maar iedereen die een computer heeft kan dit maken. Daarmee kan men leuk spelen en al doende leert men iets over deze motoren. Voor meer informatie verwijs ik graag naar de literatuur opgave.

Voor ons zeer eenvoudige zelfbouw project zijn vier zaken nodig:

a. De Motor.

Nodig is een kleine unipolaire motor, misschien is er in de rommeldoos nog wel een te vinden en ander kunt u er een kopen, kost soms f 2,50 of f 5,00 op de rommelmarkt. Een unipolaire stappenmotor heeft twee spoelen ingebouwd en iedere spoel heeft 3 aansluitingen. Totaal heeft deze motor dus 6 draden. Met de universeel meter is de weerstand te meten en welke draden aan dezelfde spoel zitten.

b. De Interface.

Hiervoor dient men het volgende schema te bouwen:

**Alle begin is moeilijk,
dus de eerste experimenten bleven heel eenvoudig**

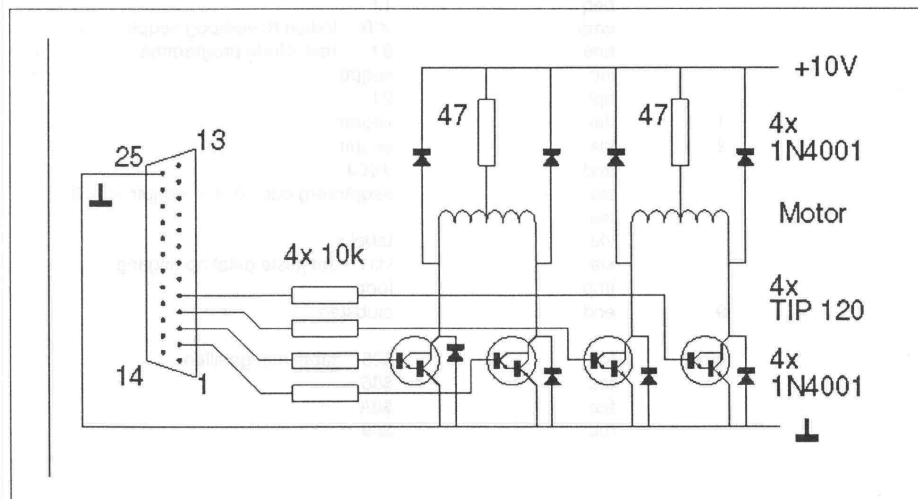


Fig. 1: schema eenvoudige stappenmotorsturing

Dat kan bijvoorbeeld op een stukje gaatjesboard.

Benodigde onderdelen:

4 x 10 kohm

2 x 47 ohm

4 x TIP 120

8 x 1N4001

1 x 25 pol. sub D.conn.

1 x stappenmotor

Opmerking: de transistoren moeten beslist darlington typen zijn, maar verder zijn de specificaties niet zo belangrijk. Ze moeten geschikt zijn voor 20 volt en 200 mA.

Waarschuwing: als u de draden van de motor verkeerd aansluit dan draait de motor straks de ver-

keerde kant op. Eenvoudig te verhelpen door de twee einden van 1 spoel te verwisselen.

c. De Computer.

Hier gaan we niet aan solderen, u kunt de parallel printer (Centronics) aansluiting gebruiken.

d. De Software.

Hieronder de meest eenvoudige software die denkbaar is om de motor te bedienen.

Het werkt zeer eenvoudig: R = 1 stapje rechtsom en L = 1 stapje linksom.

Als dit eenmaal werkt dan kunt u vermoedelijk zelf wel de software uitbreiden om de motor meer te

inch	equ	\$C020input character
loupch	equ	\$C041omzetten lower- naar uppercase
via1	equ	\$E100adres van Via1
seqptr	equ	\$300 variabele
	org	\$400
clubstep	lda	#0
	sta	via1 maak via1, poort B = 0
	lda	#\$FF
	sta	via1 + 2 maak via1, poort B = output
	lda	#0
	sta	seqptr
loop	jsr	inch input character
	jsr	loupch
	cmp	#'L indien L: verlaag seqptr
	beq	1.f
	cmp	#'R indien R: verhoog seqptr
	bne	9.f rest: einde programma
	inc	seqptr
	bpl	2.f
1	dec	seqptr
2	lda	seqptr
	and	#\$03
	sta	seqptrzorg dat: 0 <= seqptr <= 3
	tax	
	lda	tabel,x
	sta	via1 zet juiste getal op uitgang
	jmp	loop
9	end	clubstep
tabel	fcc	\$05 tabel met getallen
	fcc	\$06
	fcc	\$0A
	fcc	\$09

Fig. 2: stapjes vooruit in DOS65

laten doen. Een hele omwenteling kunt u programmeren door de stap rechts gewoon een aantal maal te herhalen. Daarbij dan wel opletten dat de software na elke stap even wacht want de computer is veel sneller dan de motor. De meeste motoren hebben toch wel 20 ms nodig om een stapje te maken.

Zie figuur 2 voor Dos65 in assembler.

En dan hier voor de PC, een programma in Borland Turbo Pascal, versie 5.5:

De eerste reactie zal zijn: met Pascal is het gemakkelijk: gewoon WRITE gebruiken om iets naar de printer poort te sturen. Op een PC kan dat echter niet, omdat de BIOS en de hardware van de PC dan verwachten dat de printer ook handshake geeft, dus Ack en Busy. De interface die hier beschreven is kan dat allemaal niet en dus moet de software ook niet moeilijk doen, maar op de meest eenvoudige manier gewoon bytes naar buiten sturen. Bij mijn PC gaat dat vanuit Pascal gelukkig met de PORT instructie. Dat is niet netjes, maar ik weet niets van PC's en vind het dus al heel fijn dat hier een uitgeprobeerd en werkend demo-programma in dit artikel staat. Als een lezer weet hoe dit beter en netter te pro-

```

program clubstep;
{ een eenvoudig programma om een stappen motor te sturen }
{ stappenmotor hangt aan de parallel printer poort , 13 jan. 1995 HS }

uses Crt;

const
  uitgang = $378; {getal evt. aanpassen aan uw computer }
  tabel: array [0..3] of integer = ( 5, 6, 10, 9 );

var
  toets: char;
  seqptr: integer;

Begin
  seqptr := 0;
  Repeat
    toets := ReadKey;
    toets := Chr( ord(toets) AND $DF ); {hoofdletter }
    write(toets);
    if toets = 'R' then inc(seqptr);
    if toets = 'L' then dec(seqptr);

    seqptr := seqptr AND $03;
    port[uitgang] := tabel[seqptr]; {getal op uitgang }

  until ( toets < > 'L' ) AND ( toets < > 'R' );

end.

```

Fig. 3: In PASCAL wil het ook

grammeren is op een PC, laat het me dan a.u.b. even weten. Hebben we gelijk iets om in het volgende blad te publiceren.

Even nog een klein beetje uitleg:

En stappenmotor is een raar soort elektromotor want de as gaat niet draaien als er gelijkspanning op de motor wordt aangesloten en ook niet met wisselspanning. Wanneer men m.b.v. elektronica de verschillende spoelen in de motor van stroom voorziet, dan draait de as van de motor steeds een klein stukje van bijvoorbeeld 7,5 graad. Een cirkel heeft 360 graden, dus de motor heeft dan 48 stapjes/omwenteling. Deze motoren bestaan in allerlei uitvoeringen: Van hele kleintjes, 2 cm, tot de grotere typen van 10 cm diameter en 20 cm lang. Sommigen hebben slechts 48 stappen per omwenteling, maar er zijn er ook van 200 stappen/omwenteling. Sommige motoren hebben twee spoelen met midden aftakking, dus totaal 6 aansluitingen (unipolaire motoren). Er bestaan er ook met 4 aansluitingen (bipolaire motoren). De elektronica om deze motoren te besturen is er ook in vele soorten, maar daarover volgende keer meer. Deze theorie was wel heel erg beknopt, maar dit was ook geen theorie verhaal.

Bij vragen: pak de telefoon of beter: kom naar de bijeenkomst en neem uw creatie mee.

Henk Speksnijder

Literatuur

Elektuur:

1985 apr, blz.28, stappenmotoren, algemeen artikel.

1985 apr, blz.54, kleine XY plotter, compleet project t.b.v. matrix printer loopwerk.

1986 jul, blz.90, stappenmotor besturing met TEA 1012 + 4 power transistoren, is een chopper, alleen voor unipolaire motoren.

1986 jul, blz.148, bipolaire stappenmotor stroomsturing m.b.v. transistors, erg beperkt en eenvoudig.

1987 jan, blz.38, universele stappenmotor-kaart deel 1

1987 feb, blz. 72, universele stappenmotor-kaart deel 2 Chopper, microstepping, uni- of bipolaire motoren. IC's: L293 (max. 1A) of L298 (max.2A) compleet project met sturing over centronics interface.

1987 jul, blz.120, eenkristals stappenmotorsturing. Chopper, voor bipolaire motoren. IC: Motorola MC3479P, max. 350 mA.

1987 okt, blz.62, mondriaan plotter deel 1

1987 nov, blz.62, mondriaan plotter deel 2 sturing via centronics, IC: MC3479P

1988 jul, blz.142, pittige stappenmotor besturing. Chopper, uni- of bipolaire motoren. IC's: L297 en L 298

1989 jul, blz.110, XY plotter interface alleen unipolaire motoren IC: ULN2803A, max. 500 mA.

1990 dec, blz.70, mondriaan plotter, software voor HPGL.

1992 jun, blz.80, pc-stappenmotor besturing-skaart uni- of bipolaire motoren, sturing met transistoren, max. 500 mA. insteek kaart voor PC.

C++ de opvolger van C (deel 2)

Inleiding

In deze tweede aflevering van de cursus gaan we ons bezig houden met een onderwerp waar Hans van Boheemen in zijn C cursus niet aan toe is gekomen. Ik ga het hebben over samengestelde datastructuren en pointers. Dat klinkt allemaal ingewikkeld en eigenlijk is dat ook zo, maar ik zal proberen het met voorbeelden toch zo goed mogelijk duidelijk te maken.

Pointers en samengestelde datastructuren zal ik uitleggen aan de hand van een wat groter programma in C++. In dit programma wordt op de computer een eenvoudige kaartenbak bijgehouden. Hoewel het zeer verleidelijk is in dit programma al gebruik te maken van de object-georiënteerd zaken die verderop in de cursus aan bod komen, zal ik dat toch niet doen. Het programma zal nog heel veel lijken op een programma in ANSI C.

Foutjes en zetduiveltjes

Bij het doorlezen van mijn verhaal in *µP Kenner* 87 viel mijn oog op een paar kleine zaken die niet helemaal goed in het blad zijn terechtgekomen. In de eerste plaats is daar figuur 1. In deze figuur heb ik getracht de verbanden tussen de diverse C-versies weer te geven. Helaas zijn de teksten in de figuur niet leesbaar. Deze zijn:

- In de binnenste ellips: K & R C (van Kernigan en Ritchie C)
- Vervolgens: ANSI C
- Dan Uitgebreid C
- En tenslotte in de buitenste cirkel: C++

De tweede fout die ik gevonden heb zit in een programma-fragment. Ik heb alle programma's daadwerkelijk door de C++ compiler laten vertalen en getest en toch is er in de listing van het programma-fragment bovenaan bladzijde 25 in de linker kolom een foutje geslopen. De eerste regel moet namelijk zijn:

```
extern "C"
```

Dus met C tussen twee dubbele quotes.

Pointers

Goed, tijd voor het echte werk. Ik heb in de vorige aflevering al aangegeven dat een pointer niets meer of minder is dan een adres. Een adres is de plaats waar een datatype in het geheugen begint. Dit adres kun je uiteraard in een variabele stoppen. Je krijgt

dan een geheugenplaats waarin het adres van een ander stuk geheugen staat. Neem het volgende voorbeeld:

```
int      *wijzer,plaats;
wijzer = &plaats;
```

In de variabele "plaats" kun je een geheel getal opslaan. De variabele "wijzer" is gedeclareerd als een verwijzing, een pointer dus, naar een int. Door middel van de tweede regel wordt het adres van "plaats" in "wijzer" geschreven. De & (= ampersand) operator geeft in dit geval aan dat in de pointer "wijzer" het adres van de variabele "plaats" wordt geschreven waarin een geheel getal kan staan. Wil je nu via "wijzer" de inhoud van "plaats" benaderen, dan gebruik je de operator "*". Je krijgt dan bijvoorbeeld:

```
*wijzer = 3;
```

Goed, nu iets serieuzer.

Hans heeft in zijn cursus in les 4 de array behandeld. Een array is een opéenvolging van dezelfde datatypes die genummerd zijn van 0 tot de lengte van de array. Bijvoorbeeld:

```
int temp[8]; // Deze array bevat 8 elementen
int *wijzer; // Deze pointer is voor uitleg
```

We hebben nu een lijstje van 8 waarden gedefinieerd. Elke individuele waarde kunnen we benaderen door de index in de array op te geven bijvoorbeeld temp[0], temp[1] etc. tot en met temp[7]. We kunnen dit echter ook doen door middel van de pointer "wijzer". Als we opgeven:

```
wijzer = temp;
```

dan wil dit zeggen dat "wijzer" gevuld wordt met het adres van het eerste element in de array "temp". De naam van een array zonder index betekent namelijk het adres van het eerste element dus wijzer = temp wil precies hetzelfde zeggen als:

```
wijzer = &temp[0];
```

en nu komt het leuke:

```
wijzer + +
```

wil zeggen: Verhoog de inhoud van het adres met het de grootte van één element in bytes. Na `wijzer++` wijst `wijzer` dus naar `temp[1]`. Dit geldt in het algemeen. Als je een pointer met een waarde verhoogt of verlaagt, dan wordt niet het adres met die waarde verhoogd of verlaagd, maar de index in de array. Het adres wordt dus met de waarde maal de afmeting van het onderliggende datatype verhoogd of verlaagd. Dus in het volgende voorbeeld:

```
wijzer = temp;
wijzer++;
```

verwijst `wijzer` na de optelling naar het element `temp[7]`. Uiteraard hoeven pointers niet altijd naar hele getallen te verwijzen. Een pointer mag naar elk datatype verwijzen. Hiervan zullen we later nog zeer veel gebruik maken. Er zijn zelfs pointers waarvoor niet is vastgelegd waar ze naar toe verwijzen. Dit zijn de void pointers waarover ik het in de vorige aflevering al even heb gehad.

Tenslotte bij deze eerste inleiding over pointers nog een klein voorbeeld waar in het gebruik van pointers hopelijk nog wat wordt verduidelijkt. Voor het gemak zijn in de listing regelnummers opgenomen. Als je het programma door de C++ compiler wilt laten vertalen, dan moeten deze regelnummers uiteraard worden weggelaten.

Het is een klein programmaatje dat de eerste tien priemgetallen afdrukt. Op regel 5 zien we een constante gedeclareerd waarmee wordt aangegeven hoeveel priemgetallen worden afgedrukt. In regel 6 wordt een array gedeclareerd waarvan de elementen meteen met de juiste waarden worden gevuld. Voor een zogenaamd scalair datatype (een getal of een letter) doe je dit door achter het `=`-teken meteen de waarde te schrijven, zoals in regel 5 is gebeurd. Array's kun je bij de declaratie vullen door de waarden, gescheiden door komma's, tussen accolades te zetten.

In regel 9 wordt de pointer "`wijzer`" gevuld met het adres van het eerste element in de array en in regel

```
1 #include <iostream.h>
2
3 void main()
4 {
5     const array_lengte = 10;
6     int priem[array_lengte] = {1,2,3,5,7,11,13,17,19,23};
7     int *wijzer;
8
9     wijzer = priem;                // Het eerste element
10    cout << "De eerste " << array_lengte
11         << " priemgetallen zijn \n";
12
13    for (int i = 0; i < array_lengte; i++)
14        cout << i << "\t" << *wijzer++ << "\n";
15
16    cout << "\nEn nu van achteren naar voren \n";
17    wijzer = priem + array_lengte - 1;    // Het laatste element
18
19    for (i = 0; i < array_lengte; i++)
20        cout << i << "\t" << *wijzer-- << "\n";
21
22    cout << "\n";
23    return;
24 }
```

Fig. 1: voorbeeld van het gebruik van pointers

17 met het adres van het laatste element. Je ziet in regel 17 dat we ons helemaal niet druk maken over de grootte van het datatype waarnaar wijzer verwijst; we zeggen gewoon dat de pointer moet wijzen naar het element dat 9 plaatsen (10 - 1) verder ligt dan het eerste element.

In regel 13 en 14 staat een lus die tien maal doorlopen wordt (de teller *i* loopt van 0 t/m 9). In deze regel is gebruik gemaakt van het feit dat je vrijwel overal nieuwe variabelen mag declareren, bijvoorbeeld ook in de definitie van een for-lus. In regel 14 staat iets dat typisch C of C++ is. In deze regel wordt het getal waarnaar de pointer wijzer verwijst afgedrukt en wordt vervolgens de pointer één plaats opgeschoven. Met name dit soort afkortingen maakt de taal voor mensen die met C++ beginnen behoorlijk ondoorzichtig. Voor luie programmeurs heeft C++ de mogelijkheid dergelijke dingen kort op te schrijven; mensen die daar niet zo vertrouwd mee zijn, kunnen van de opdracht binnen de lus ook twee afzonderlijke opdrachten maken die dan uiteraard wel tussen accolades moeten staan.

Voor extreem luie programmeurs kunnen de regels 9 en 13 resp. 17 en 19 ook nog worden samengevoegd. De definitie van een for-lus bestaat namelijk uit drie delen die gescheiden worden door een `;`. Dit is een deel dat uitgevoerd wordt bij het begin van de for-lus (int *i* = 0), het deel waarin wordt bepaald of de for-lus beëindigd wordt (*i* < array_lengte) en het deel dat aan het einde van elke keer dat de for-lus doorlopen wordt uitgevoerd wordt (*i*++). Nu mag je in deze definitie in elk deel ook meerdere opdrachten plaatsen die dan door middel van een `,` van elkaar gescheiden worden. Je mag dus schrijven:

```
for (i = 0, wijzer = priem;
    i < array_lengte; i++, wijzer++)
    cout << i << "\t" << *wijzer << "\n";
```

Op dat moment is het niet meer mogelijk de declaratie van *i* (int) in de for-lus op te nemen. Deze moet dus aan regel 6 of 7 worden toegevoegd.

Klassen en structuren

Een tweede zaak die Hans niet heeft behandeld zijn de zogenaamde gestructureerde datatypes of structuren. Omdat er in C++ vooral in deze groep de belangrijkste verschillen zitten met ANSI C, komen we er in deze serie niet omheen hier uitgebreid op in te gaan.

Elke programmeertaal heeft een aantal datatypes. Dat zijn in de eerste plaats uiteraard gehele getallen (integers), drijvende komma getallen en (letter-) tekens. Van deze types is precies bekend hoe groot ze zijn. ANSI C en C++ hebben naast deze types ook nog een aantal andere types waarvan de afmeting niet bij voorbaat vastligt. Hiervan heeft Hans de string als een verzameling van tekens en de array behandeld. In deze paragraaf wordt hier de klasse aan toegevoegd. Nu is klasse (of class) een typische C++ benaming. In ANSI C spreekt men van een structure of struct. In C++ is er een (klein) onderscheid tussen een class en een struct. We beginnen met het behandelen van de struct zoals die ook in ANSI C voorkomt.

Om duidelijk te maken wat met een struct bedoeld wordt, gaan we maar meteen kijken naar een praktijkvoorbeeld. Ik heb naast de telefoon een kaartenbakje met daarin een aantal kaarten waarop de gegevens van onze familie en kennissen staan. Elke gezin heeft een kaartje met daarop de volgende gegevens:

- Achternaam gezinshoofd
- Voornamen gezinshoofd
- Achternaam partner
- Voornamen partner
- Adres en huisnummer
- Postcode
- Woonplaats
- Telefoonnummer

En nog een aantal gegevens die vaak weer afhankelijk zijn van de situatie (wel/geen kinderen, familielid/collega/clubgenoot, geboortedata, gironummer etc.). Het kaartje met mijn eigen gegevens zit niet in de bak, maar als dit wel het geval zou zijn, dan ziet deze er als volgt uit:

- Van Opbroek, Gerrit
- Jansen, Lambertina Anna Hermina
- Den Del 16
- 5071 TT Udenhout
- 04241-3795

Deze gegevens horen duidelijk bij elkaar en in mijn kaartenbak zit één kaartje met daarop deze gegevens. Toch heeft elk kaartje eenzelfde structuur. Precies hetzelfde kunnen we doen in C++ of ANSI C. Hier kunnen we een structuur definiëren voor de bovengenoemde gegevens:

```

struct kaartje
{ char    achternaam_gezinshoofd[30],
  voornamen_gezinshoofd[30],
  achternaam_partner[30],
  voornamen_partner[30],
  adres[30],
  postcode[8],
  woonplaats[30],
  telefoon[15];
};

```

Op deze manier is vastgelegd welke informatie er op een kaartje vastgelegd kan worden. Let op, op deze manier is alleen nog maar vastgelegd wat er op een kaartje komt, er is nog geen kaartje gedeclareerd. In de structuur zijn alleen maar strings vastgelegd. Dat is toeval, in een structuur mogen alle bekende datatypes, dus ook bijvoorbeeld integers, array's, en zelfs reeds gedefinieerde structuren worden gebruikt. We hadden dus ook kunnen zeggen:

```

struct persoon
{ char    achternaam[30],
  voornamen[30];
  int     geboortedag,
  geboortemaand,
  geboortjaar;
};

struct kaartje
{ struct persoon    gezinshoofd,
  struct persoon    partner,
  int     kinderen[5];
  char    adres[30],
  postcode[8],
  woonplaats[30],
  telefoon[15];
};

```

We hebben nu op het kaartje ook nog ruimte gemaakt voor de geboortedata en voor vijf kinderen (genummerd van 0 t/m 4). Als we nu in een programma deze gegevens willen gebruiken, dan moet er voor elk kaartje ruimte worden gereserveerd. Dat kan op meerdere manieren, maar laten we eerst de meest eenvoudige manier behandelen, we declareren een variabele van het type kaartje en met de naam gert:

```

kaartje    gert;

```

In het geheugen wordt dan een stuk geheugen gereserveerd ter grootte van het type kaartje. Dit geheugen is bereikbaar onder de naam gert. Stel nu dat we het adres van gert willen benaderen. Dit gaat heel simpel zoals bijvoorbeeld blijkt uit:

```

strcpy(gert.gezinshoofd.achternaam,"Opbroek van");
strcpy(gert.adres,"Den Del 16");
waarmee mijn adres naar het veld adres in de variabele gert wordt gekopieerd en mijn naam naar het veld achternaam in de structuur gezinshoofd in de variabele gert.

```

Zoals reeds is behandeld, kun je pointers naar alle datatypes maken; dus ook naar structuren. Je kunt dus zeggen:

```

kaartje    *wijzer,gert;

```

```

wijzer = gert;

```

Als je nu via de pointer wijzer het veld postcode wil benaderen, dan schrijf je:

```

*wijzer.postcode .....

```

Dit kan echter ook afgekort (nou ja) worden tot:

```

wijzer->postcode .....

```

Hoewel dit niet echt een afkorting is, verdient de laatste notatie toch de voorkeur omdat die over het algemeen wat leesbaarder wordt gevonden, zeker bij de ingewikkelde structuren waar we later in de serie nog uitgebreid kennis mee zullen maken.

Ik heb reeds verteld dat er in C++ een verschil is tussen een class en een struct. Dit verschil bestaat hieruit, dat zonder verdere maatregelen van een struct alle elementen van buiten af benaderd kunnen worden en van een class geen enkele. Dit heeft alles te maken met de object-oriëntatie die in de taal C++ zit waarbij een klasse bepaalde elementen kan verbergen voor de buitenwereld. Op deze interessante onderwerpen kom ik in een volgende aflevering uitgebreid terug. Vooralsnog maken we alleen gebruik van de struct zodat we ons daarover nog geen zorgen hoeven te maken.

Unions en bitfields

Het komt in de praktijk ook wel eens voor dat niet altijd dezelfde velden op een kaartje staan. Neem als voorbeeld weer de persoonsgegevens. Naast kaartjes met familie, kennissen etc. zijn er ook kaartjes met daarop verzekeringsgegevens. Op deze kaartjes staat dan de naam van de verzekeringsmaatschappij, de naam, het adres en het telefoonnummer van de agent en het polisnummer. Toch hebben we maar één soort kaartjes in de bak. Kunnen we in C++ iets dergelijks ook regelen?

Welnu, dat kan. We kunnen de extra gegevens natuurlijk gewoon in de structuur voor kaartje op-

nemen, maar dat is niet helemaal de bedoeling. In dat geval worden alle kaartjes namelijk veel te groot. De kaartjes van de kennissen etc. hebben overbodige velden voor de verzekerings-gegevens en de kaartjes voor de verzekeringen hebben overbodige velden met geboortedata etc.

Om dit op te lossen, bestaat er in C++ en in ANSI C de UNION. Deze datastructuur lijkt heel veel op een struct, maar heeft als kenmerk dat de velden niet achter elkaar in het geheugen liggen, maar over elkaar heen. De afmeting van de structuur is dan even groot als de afmeting van het grootste veld. Een voorbeeld van een dergelijke constructie is de volgende:

union intflo

```
{int      i;
 float    x;
} u;
```

Hiermee wordt een variabele u gedeclareerd die zowel een integer kan zijn als een float. Benaderen we de variabele met u.i, dan bedoelen we dat we de variabele als integer willen beschouwen; schrijven u.x, dan is het een float.

Laten we nu het kaartenbak-probleem eens oplossen. Dat doen we als volgt: We definiëren naast de structuur "kaartje" ook een structuur "verzekering". In deze structuur nemen we de gegevens op die op het verzekering-kaartje staan dus:

```
struct verzekering
{ char    maatschappij[30];
  char    persoon    agent;
  char    adres[30],
    postcode[8],
    woonplaats[30],
    telefoon[15];
  char    polisnummer[15];
};
```

In dit geval gebruiken zullen we van de agent de geboortedatum niet kennen. We zullen dit dus maar niet invullen. Er tientallen andere manieren om een dergelijke structuur op te bouwen, sterker nog, C++ heeft binnen het object-georiënteerde deel nog mogelijkheden die ons nu zeer goed van pas kunnen komen. Om niet teveel nieuwe dingen tegelijk in te voeren, doen we het in dit voorbeeld even op de ANSI C manier.

Als we nu een union definiëren van de structuren "kaartje" en "verzekering", zijn we waar we wezen willen:

union algemeen

```
{ kaartje    kennis;
  verzekering verzekering_gegevens;
};
```

Een onderwerp die bij de KGN absoluut niet mag ontbreken zijn de zogenaamde bitfields of bitvelden. Van een lid van onze club mag je verwachten dat ze weten dat het kleinste element in het geheugen van een computer niet een byte is maar dat deze is opgebouwd uit acht bits. Binnen C++ (en ANSI C) is het mogelijk binnen een structuur voor elk (integer) element precies op te geven uit hoeveel bits een element bestaat. Je doet dit als volgt:

```
struct bitvelden
{ unsigned vier_bits:4,
          twee_bits:2,
          drie_bits:3;
  char    c;
};
```

De velden in deze structuur kunnen nu op dezelfde manier benaderd worden als de velden binnen een normale structuur. Hierop is één uitzondering, je mag met de &-operator niet het adres van een dergelijk bitveld opvragen omdat de kleinst-adresseerbare eenheid binnen een computer over het algemeen één byte is.

Dynamische datastructuren

Zo, we hebben nu al heel wat behandeld. We hebben pointers, structuren, unions en bitvelden. In deze paragraaf gaan we dat allemaal door elkaar gooien waarna we dynamische data-structuren krijgen.

Tot nu toe hebben we in de voorbeelden steeds een pointer naar een structuur gedeclareerd en bovendien een variabele van de datastructuur (wijzer en gert). Vervolgens hebben we de pointer gevuld met het adres van de variabele om zo een datastructuur via een pointer te kunnen benaderen. Dit is echter helemaal niet nodig! Het grote nadeel is namelijk dat we in dat geval al van te voren vast moeten gaan leggen hoeveel variabelen we van dat type willen gebruiken (bijvoorbeeld door er een array van te maken). Wat we veel liever zouden doen is "naar behoefte" variabelen van het juiste type aan te maken.

Binnen C++ zijn hiervoor twee manieren. Voor beide manieren is het nodig dat er een pointer is gedeclareerd van het juiste type. De eerste manier is de manier die afkomstig is uit ANSI C. Deze manier maakt gebruik van de functie malloc waarvoor de header file <stdlib.h> nodig is. In dat geval maken we een stukje geheugen op de volgende manier beschikbaar:

```
wijzer = (kaartje *) malloc(sizeof kaartje);
```

Met deze opdracht worden evenveel bytes gereserveerd als er voor de datastructuur "kaartje" nodig zijn. Dit aantal wordt bepaald door de sizeof-operator. Voor de aanroep van de functie staat een zogenaamde typecast-operator die er voor zorgt dat de pointer die het resultaat is van malloc (dit is een void-pointer, dus alleen maar een pointer) omgezet wordt in een pointer naar een structuur van het type kaartje.

Een tweede en binnen C++ een veel betere manier is gebruik te maken van de operator new. Dit gaat als volgt:

```
wijzer = new kaartje;
```

Als we het geheugen dat we op de bovenstaande manier hebben gereserveerd weer vrij willen geven, dan doen we dat op de volgende manier:

```
free(wijzer);
```

resp.

```
delete wijzer;
```

Nu kan het voorkomen dat het systeem bij het reserveren van geheugen tot de conclusie komt dat er onvoldoende vrij geheugen beschikbaar is om aan de vraag te voldoen. In dat geval krijgt de pointer niet het adres van het blok geheugen als waarde maar de waarde null. Deze waarde is gelijk aan het getal 0 maar het is beter het woord 'null' te gebruiken. Wel is het dan noodzakelijk de header file <stddef.h> op te nemen. Als algemene regel kan men stellen dat een pointer die niet ergens naar toe wijst, de waarde null behoort te hebben. In een aantal gevallen zal dit automatisch gebeuren (gedeclearde variabelen, dus ook pointers worden automatisch op 0 geïnitieerd).

Voor de volledigheid wordt verder nog opgemerkt dat ANSI C en dus ook C++ naast de genoemde functies malloc en free ook nog de functies calloc (doet vrijwel hetzelfde als malloc) en realloc (vergroot een datastructuur met een aan te geven hoeveelheid bytes) kent. In deze cursus worden deze functies verder niet beschreven.

Dynamische datastructuren zijn heel belangrijk. Je kunt namelijk tijdens het draaien van een programma bepalen hoeveel ruimte je voor gegevens nodig hebt en alleen die hoeveelheid ruimte reserveren. Ik kom hier in een uitgebreid voorbeeld nog op terug.

Parameter-overdracht en de referentie-operator

Het laatste onderwerp dat ik in deze aflevering nog wil behandelen is de overdracht van parameters naar functies. Hans heeft in de laatste aflevering van zijn C cursus dit onderwerp gedeeltelijk behandeld. In zijn voorbeelden werd een functie als volgt gedefinieerd:

```
int produkt(int a,b)
{ return a*b;
}
```

In deze functie komen variabelen a en b voor die bij de aanroep van produkt gevuld worden. Verder geeft de functie het produkt van a en b als resultaat terug. Bij deze constructie wordt bij de aanroep van produkt de waarde van a en b ingevuld. Je mag dus ook schrijven:

```
c = produkt(2+3,4);
```

waarna c de waarde 20 krijgt. Dit wordt "call by value" genoemd. Zouden we een variabele als parameter meegeven, dan wordt de waarde van die variabele doorgegeven aan de functie. Als in de functie de variabele die met de parameter a wordt aangeduid wijzigt, dan wordt deze wijziging niet mee teruggegeven. Probeer maar:

```
#include <iostream.h>

void produkt(int a,int b,int c)
{ c = a * b;
  return;
}

void main()
{ int c = 10;

  produkt(2,4,c);
  Cout << "De waarde van c = "<< c << "\n";
  return;
}
```

De variabele c wordt niet door de functie produkt teruggegeven. In ANSI C is de oplossing hiervoor niet c zelf als parameter door te geven maar het adres van c. In dat geval moet je de parameter als pointer declareren:

```
#include <iostream.h>

void produkt(int a,int b,int *c)
{ *c = a * b;
  return;
}

void main()
{ int c = 10;
```

```

produkt(2,4,&c);
cout << "De waarde van c = " << c << '\n';
return;
}

```

Je geeft nu in de aanroep door middel van de &-operator (referentie-operator) aan dat het adres van de variabele *c* moet worden doorgegeven. Dit is zuiver theoretisch nog steeds een call by value.

In C++ hebben we naast de call by value ook de zogenaamde call by reference. In dat geval wordt in de functie aangegeven dat de parameters referentie-parameters zijn. Het geheel ziet er dan als volgt uit:

```

#include <iostream.h>

void produkt(int a,int b,int &c)
{ c = a * b;
  return;
}

void main()
{ int c = 10;

  produkt(2,4,c);
  cout << "De waarde van c = " << c << '\n';
  return;
}

```

In dit geval wordt de referentie-operator in de functie-definitie opgenomen. Deze referentie-operator

kan op nog veel meer manieren worden toegepast. Ik kom daar in de volgende aflevering nog op terug.

Afsluiting

Dit is een vrij omvangrijke aflevering geworden. Bij deze aflevering hoort nog een uitgebreid voorbeeld waarin een eerste begin wordt gemaakt met een kaartenbakprogramma. Dit programma is elders in dit blad te vinden.

De volgende keer gaan we o.a. uitgebreider in op de stream I/O. Bovendien gaan we nog wat dieper in op het gebruik van functies binnen een C++ programma.

Tot dan.

Get van Opbroek

Referenties:

- 1: Leen Ammeraal: C++ 2e herziene druk; Academic Service (1991)
- 2: Turbo C++ User's Guide; Borland 1991
- 3: Turbo C++ Programmer's Guide; Borland 1991
- 4: Heiner Högel: C plus einhalb, Umstieg von C auf C++; DOS International 7/94
- 5: Dirk Schaun: C++ - eine Sprache mit Know-how; DOS International 8/94
- 6: Hans van Boheemen: Cursus C; µP Kenner 81 t/m 86
- 7: Jack Purdum, Timothy C. Leslie: C Standard Library; Que 1987

The Ultimate

The BBS for all systems




Telefoon nummers::

Tel:: 053-303902 (2lijnen) V32b / V42b / V.Fast
 053-327457 V32b / V42b
 053-311799 V32b / V42b / 16k8HST
 053-312006 V32b / V42b / 14k4HST
 053-328506 ISDN

Snelheden :: V22 - V22bis - **V32bis** - V42bis

Beweging, deel 2

In deel 1 is een interface en een zeer eenvoudig programma beschreven. Voor het volledig besturen van stappen motoren is echter meer nodig, maar het lijkt me niet zo handig om een listing van het programma te publiceren. In plaats daarvan is het beter om enkele aspecten van de software te bespreken.

De Bits op de uitgang:

Een interface voor een stappenmotor heeft meestal 4 aansluitingen. Om een stappenmotor te laten bewegen is het noodzakelijk de aansluitingen in een bepaalde volgorde te besturen. Het benodigde bit-patroon hangt af van de interface en van de gebruikte stappenmotor.

Met de interface van deel 1, bijvoorbeeld:

STEPTABEL volstap:

0101	spoelen	A +	B +
0110	spoelen	A +	B -
1010	spoelen	A -	B -
1001	spoelen	A -	B +

STEPTABEL half stap:

0101	spoelen	A +	B +
0100	spoelen	A +	
0110	spoelen	A +	B -
0010	spoelen		B -
1010	spoelen	A -	B -
1000	spoelen	A -	
1001	spoelen	A -	B +
0001	spoelen		B +

Voor de eenvoud ga ik niet proberen om met de software dit rare patroon te berekenen, maar zet gewoon dit bit-patroon in een tabel. Dan moet de software alleen een teller in deze tabel bijhouden. De teller heet: seqptr en om de motor rechtsom te besturen is dan nodig:

```
seqptr = seqptr + 1;
IF seqptr = 8 THEN seqptr = 0;
OUTPUT(steptabel[seqptr]);
```

En linksom is dan natuurlijk:

```
IF seqptr = 0 THEN seqptr = 8;
seqptr = seqptr - 1;
OUTPUT(steptabel[seqptr]);
```

Om de software universeel te houden, gebruik ik altijd een tabel van 8 bytes. Bij volstap methode bestaat die tabel dus uit 2x dezelfde 4 bytes. Het voordeel van deze tabel is, dat men bijv. bij een andere hardware, alleen die tabel moet veranderen en niet het hele programma.

Met 2 motoren:

Voor de meeste apparaten zoals plotters, draaibanen en freesbanken zijn 2 motoren nodig. De ene motor noem ik X-motor want die beweegt de x-as van de machine, logisch dat de andere motor dan de Y-motor is. Omdat iedere motor 4 draden heeft, kan dit met een 8-bit output poort. Bij DOS65 kan de Via2, Poort A gebruikt worden. De X-motor is verbonden met bits 0, 1, 2 en 3 en de Y-motor met bits 4, 5, 6 en 7. Om dat te besturen zijn natuurlijk 2 tellers nodig: seqptrx en seqptry. Het byte dat naar de uitgang moet is

dan:

```
A = steptabel[seqptry];
A = 16 * A;
A = A + steptabel[seqptrx];
output(A);
```

De richting:

Er is nog een klein probleempje dat men bij twee motoren in 8 verschillende richtingen kan bewegen, bijvoorbeeld X-motor rechtsom en Y-motor ook, of

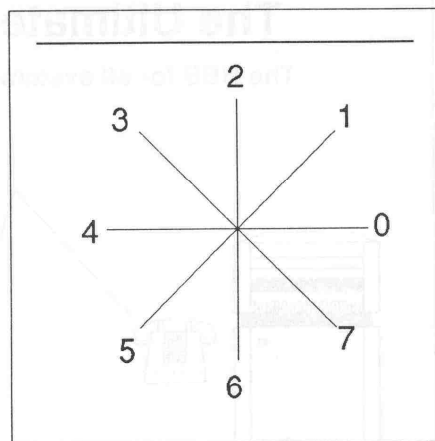


Fig. 1: de staprichtingen

motor rechtsom en Y-motor staat stil enz. Om de rest van het programma eenvoudig te houden, is er een subroutine gemaakt welke heet STAP. Die maakt 1 stapje en de accu bevat een getal 0..7 dat aangeeft in welke richting de stap moet:

De bewegingen:

Een rechte lijn is nu niet zo moeilijk meer, bijv 300 stapjes naar links:

```
FOR i=0 to 300 DO stap(4);
```

De tijd/stap:

Bij het besturen van een plotter is het niet zo belangrijk hoe snel (of hoe traag) de pen beweegt, maar bij een draaibank is dat wel heel belangrijk. Daarbij is het vooral van belang dat de beweging zeer regelmatig is. Anders is op het werkstuk precies te zien waar de machine even stil gestaan heeft. Voor deze zeer regelmatige beweging is het nodig dat we precies weten wat de CPU in onze computer zoal te doen heeft.

In DOS65 is er buiten het user-programma alleen nog een clock- en evt. een RTC-interrupt. Die kunnen gewoon uitgezet worden. De rest van de computer heeft daar totaal geen last van, alleen gaat de systeem-tijd dan achter lopen. Wanneer u echter een RTC heeft, dan komt alles vanzelf weer goed, als de clock weer aangezet wordt.

Om de tijd per stap goed te regelen op DOS65 gebruik ik de VIA want die heeft twee ingebouwde timers. De ene timer dient om (uit de systeemklok) een frequentie van 10 kHz te maken op uitgang PB7. Op de print met een jumper PB7 en PB6 verbinden. De andere timer gebruikt dan die 10 kHz om af te tellen en naar nul. In die tweede timer wordt dan een getal gezet dat bepaald hoe lang een stap moet

duren, op 0,1 ms nauwkeurig. De langste tijd is dan ongeveer 6 seconden/stap en daarmee is het regelbereik zo groot dat er elke soort stappenmotor of machine mee te bedienen is. Het is bij DOS65 heel belangrijk dat de timer vanzelf doortelt als de CPU aan het rekenen is want de 6502 is net snel genoeg en de rekentijd naar de volgende stap is beslist niet constant. Het systeem is dan als volgt:

- Start timer
- Bereken volgende stap
- Wacht tot timer op nul staat
- Zet nieuwe byte op uitgang

In de meeste andere computers kunt u beter niet met die zaken als systeem clock en interrupts gaan rommelen en maar hopen dat een moderne 32-bit CPU zo snel is, dat u van een interrupt meer of minder niets merkt. Het is wel belangrijk om dit soort besturingen niet in een multi-tasking systeem te draaien want dan kan de timing wel een groot obstakel vormen. Tenzij u die problemen oplost met een speciale interface natuurlijk.

Hoe men met een PC een timing kan regelen die zo fraai instelbaar is als die van DOS65, dus op 0,1 ms nauwkeurig en instelbaar tussen 1 ms en 1000 ms weet ik niet. Kan hier iemand helpen? Voorlopig gebruik in een nood-oplossing: in Borland Turbo-C zit een functie die heet DELAY en daarmee is een vertraging tussen 1 en 1000 milliseconden goed te maken. Het nadeel is dat het programma eerste gaat rekenen en daarna de delay doet. Zodat vooral bij snelle bewegingen de tijd/stap erg afhankelijk is van de rekentijd. En de kleinste tijd is 1 ms zodat er weinig keus is, het is 2 of 3 en 2,5 ms kan dus niet.

Berekeningen:

Rechte bewegingen langs de X-as of langs de Y-as zijn nu gemakkelijk, maar dat kon de freesbank ook zonder die stappenmotoren al. Deze hele toestand wordt pas nuttig als de machine dankzij de computer besturing ook

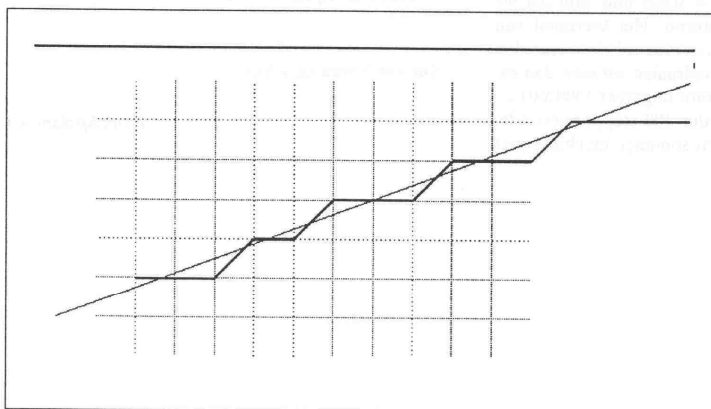


Fig. 2: trapjes in plaats van een lijn

schuine lijnen en cirkels kan maken. Daarvoor is nog wel een heel programma nodig. Voor een schuine lijn welke niet onder 45 staat moet de computer een soort verdeling maken van schuine en rechte stappen, zodanig dat het gereedschap uiteindelijk ongeveer de gewenste beweging maakt.

In figuur 2 is een poging gedaan om het te tekenen: de gewenste schuine lijn en wat voor soort schuine lijn we kunnen maken met een machine die bestuurd is met stappen motoren:

Het zal duidelijk zijn dat de problemen bij een cirkel nog wat moeilijker zijn. Zeker als de berekeningen moeten worden uitgevoerd met een 6502. Hoe dit kan, zal ik u besparen, want dan wordt het verhaal wat moeilijk. U hoeft ook niet alles te programmeren want de auteur heeft op DOS65 een werkend programma in assembler. Speciaal voor degenen die een andere computer hebben, wordt dat nu opnieuw geschreven, bijna helemaal in Ansi-C. Voor vrijwel elke computer is er een C-compiler beschikbaar, dus daarom is die taal gekozen. Er zijn enkele speciale functies nodig welke niet in Ansi-C bestaan en die dus voor elk type computer speciaal gemaakt moeten worden. Voor de ibm-klonen en voor de Acorn 5000 is het C-programma direct te gebruiken. Alle speciale instructies voor die machines zitten er al in.

De rekentijd:

Het zou allemaal niet zo moeilijk zijn, als er ruimschoots genoeg tijd is, maar helaas is er niet veel tijd om alle berekeningen te doen. Op de meeste machines wil men toch zeker op 0,01 mm kunnen instellen. Het kan best dat door de mechanische constructie men uitkomt op 1 stapje = 0,005 mm. Stel dat we een draaibank willen besturen. Het toerental van een eenvoudige machine kan meestal niet meer dan 1500 omw/min en bij het nadraaien wil men dan ca. 0,02 mm/omw. zodat de motor ongeveer $1500 \times 0,02 / 60 = 0,5$ mm/seconde en dus 100 stappen/seconde. Bij het voordraaien kunnen sommige machines met

1000 omw/min en 0,5 mm/omw werken en dat wordt dan $1000 \times 0,5 / 60 = 8$ mm/s en dus 1600 stappen/seconde. Een stappenmotor die 1000 stappen/sec kan is al een hele snelle motor. We willen uiteraard dat de stappen motor dan de beperkende factor in de machine is en niet de computer. Daarom moeten we uitgaan van een rekentijd van 1ms/stap.

Een kleine machine heeft bijv. een lengte heeft van 500 mm en met 100 stappen/mm moeten we dus max. 50.000 stappen maken voor de hele beweging. Aangezien we in de software moeten bijhouden waar we gebleven zijn in het werkbereik van de machine is daarvoor een teller nodig die minimaal tot 50.000 kan tellen.

Er zijn ook machines van 700 mm lang en misschien wel 400 stappen/mm. In sommige bewerkingen kan het voorkomen dat de machine een gedeelte van een cirkel moet maken waarbij het middelpunt van de cirkel buiten het werkbereik licht. Dat redden we niet met een 16 bit getal. Eigenlijk willen we dus gewoon dat de computer met getallen overweg kan die aanzienlijk groter zijn dan 65.000.

Er is 1 troost: alle rekenwerk kan met integer getallen (gehele getallen, dus geen komma) omdat de stappen motoren allen maar kunnen stappen. Een 0,3 stap kan gewoon niet dus die hoeven we ook niet te berekenen.

Uiteindelijk volgt hieruit een eisenpakket voor de reken software:

- Doe de berekeningen met getallen van 32 bit. Groot genoeg voor elke machine.
- Maak de software zodat alle rekenwerk in 1 ms kan. Ook bij het berekenen van een cirkel.

Dit was 't voor deze keer!

Henk Speksnijder

Van de Voorzitter

Opnieuw een dunne μ P Kenner, jammer maar geen kan ook nog. Een Uitnodiging voor de bijeenkomst van 22 april zou eigenlijk beter geweest zijn. Die had namelijk wel op tijd kunnen zijn.

Ondertussen is het zover dat de drie overgebleven bestuursleden het niet bol kunnen werken. Voor mij zelf zijn er de afgelopen maanden nogal al wat dingen in de privé-sfeer gebeurd. Details zijn overigens op verzoek verkrijgbaar. In die tijd behoorde KGN(68k) niet of nauwelijks tot zaken waar ik tijd voor had.

Eerder hebben wij jullie al geraagd voor versterking van het bestuur of goede ideeën. Een van de weinige ideeën, bijeenkomst in April, uitnodiging in Maart, heb ik om bovengenoemde reden niet uit kunnen voeren. Het bestuur

zal voorlopig niet versterkt worden, tenzij er nu een paar personen opstaan.

Vooraf nieuwe bestuursleden zijn (nog steeds!) hard nodig.

Maar dat lijkt mij onwaarschijnlijk. Daarom is het idee terug om de KGN activiteiten die zogoed als alleen op het BBS The Ultimate plaatsvinden. Inschrijving KvK, Postbus en dergelijke blijven dan gewoon bestaan. En als er dan een groter bestuur is dan kunnen die daar mee verder gaan.

Ik heb nogal eens een keer over stuurlui aan de wal gesproken. Deze keer wil ik ook afsluiten van een zeevaartsterm. Hij wordt gebruikt als het de laatste keer KAN zijn dat men contact met elkaar heeft: Vaarwel.

Jullie Voorzitter.

Voortgang KGN68k (deel 18)

Deze keer is er nauwelijks voortgang te melden.

Met het enige dat beter is geworden is helaas niets mee gedaan. Tijdens de HCC-dagen hebben zich namelijk drie PCB-lay-outers gemeld. Door omstandigheden bij mijn huidige werkgever ben ik er niet aan toe gekomen om deze mensen op gang te helpen. Op de andere plaatsen binnen de werkgroep

ben ik er ook niet aan toe gekomen om het vuur brandend te houden.

KGN68k wil ik best wel weer aan het rollen brengen, maar eerst weer de vaart in mij zelf brengen.

Geert Stappers

Informatie

De μ P Kenner (De microprocessor Kenner) is een uitgave van de KIM gebruikersclub Nederland. Deze vereniging is volledig onafhankelijk, is statutair opgericht op 22 juni 1978 en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305. Het gironummer van de vereniging is 3757649.

De doelstellingen van de vereniging zijn sinds 1 januari 1989 als volgt geformuleerd:

- Het vergaren en verspreiden van kennis over componenten van microcomputers, de microcomputers zelf en de bijbehorende systeemsoftware.
- Het stimuleren en ondersteunen van het gebruik van micro-computers in de meer technische toepassingen.

Om deze doelstellingen zo goed mogelijk in te vullen, wordt onder andere 5 maal per jaar de μ P Kenner uitgegeven. Verder worden er door het bestuur per jaar 5 landelijke bijeenkomsten georganiseerd, beheert het bestuur een Bulletin Board en wordt er een softwarebibliotheek en een technisch forum voor de diverse systemen in stand gehouden.

Landelijke bijeenkomsten:

Deze worden gehouden op bij voorkeur de derde zaterdag van de maanden januari, maart, mei, september en november. De exacte plaats en datum worden steeds in de μ P Kenner bekend gemaakt in de rubriek Uitnodiging.

Bulletin Board:

Voor het uitwisselen van mededelingen, het stellen en beantwoorden van vragen en de verspreiding van software wordt er door de vereniging een Bulletin Board (BBS) beschikbaar gesteld.

De telefoonnummers zijn: 053-303902, 053-327457, 053-311799 of 053-312006

ISDN: 053-328506.

Internet : kgn@ultima.iaf.nl

Software Bibliotheek en Technisch Forum:

Voor het beheer van de Software Bibliotheek en technische ondersteuning streeft het bestuur ernaar zgn. systeemcoördinatoren te benoemen. Van tijd tot tijd zal in de μ P Kenner een overzicht gepubliceerd worden. Dit overzicht staat ook op het BBS.

Correspondentie adres

Alle correspondentie betreffende verenigingszaken kan gestuurd worden aan:

KIM Gebruikersclub Nederland
Postbus 1336
7500 BH Enschede

Het Bestuur

Het bestuur van de vereniging wordt gevormd door een dagelijks bestuur bestaande uit een voorzitter, een secretaris en een penningmeester en een viertal gewone leden.

Geert Stappers (Voorzitter, KGN/68k coördinator)
Engelseweg 7
5825 BT Overloon
Telefoon 04781-41279
Internet: stappers@knowarc.nl

Penningmeester: vacant

Secretaris: vacant

Jan Veninga
Klimopstraat 51
7601 SJ Almelo
Telefoon 0546-827910

Henk Speksnijder
Albert Cuijpsstraat 43
2902 GA Capelle aan den IJssel
Telefoon 010-4586879

Redactie μ P Kenner: vacant

tijdelijk: Nico de Vries
Van der Waalsstraat 46
2984 EP Ridderkerk
Telefoon 01804-29207

Ereleden:

Naast het bestuur zijn er een aantal ereleden, die zich in het verleden bijzonder verdienstelijk voor de club hebben gemaakt:

Erevoorzitter:

Siep de Vries

Ereleden:

Mevr. H. de Vries-van der Winden
Anton Müller
Rinus Vleesch Dubois