Z80-NYT

9. ÅRGANG - NR. 1. 1988

<><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

Hexadecimal ADDITIONS-TABEL

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10
2	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
3	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12
4	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
5	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14
6	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15
7	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16
8	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17
9	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18
A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19
B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A
C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Z80-NYT Nr. 1. 9.ÅRGANG - MARTS / APRIL 1988

UDGIVES AF Z80-BRUGERGRUPPEN, Nørre Farimagsgade 1, 1364 Kbh. K.

Bestyrelsen:

Formand	Kurt Neierdi,	Nørre Farimagsgade 1, 1364 Kbh. K.	
Tlf.tid	(mandag + torsdag)	01 11 38 94 (1700 - 1800)	
Næstform.	Frank Damgaard,	Kastebjergvej 26A	2750 Ballerup
Kasserer	- -	02 97 37 47 (1730 - 1830)	
	(efter 10-7-1988	skriftligt - tak)	
Bestyr.m.	Carsten Busk Senholt,	Blommevangen 6/1	2765 Smørum
-	Anders Otte,	Frederiksdalsvej 40/2/th	2830 Virum
-	Henrik Jensen,	Markskellet 6/ST/TV	2720 Vanløse

CP/M Bibliotek:	pt. Kurt Neierdi	se adr. ovenfor.	(skriftligt)
Forretn.f:	pt. Frank Damgaard,	Indkøbsforeningen.	
Redaktion:	Z80-NYT,	Fensmarks Alle 6	3520 Farum
Logo:	C. B. Senholt.		
Redaktør:	pt. Viggo Jørgensen.		

Private annoncer gratis for medlemmer.

Kontingent årligt: 300 kr.

GIRO 6 74 26 02, Z80-Brugergruppen, c/o Frank Damgaard, (se adr. ovenfor).

Der køres modemforsøg med data-overførsel på telf. 01 91 38 98, der kan kaldes hele døgnet. 'Linie-opsætning' er 1200 / 1200 fuld duplex 1 startbit, 8 databit, 1 stopbit, ingen paritet.

Aftal nærmere på telf. 01 11 38 94 helst i kontortiden.

Formanden.

- = o = -

Z80 Brugergruppen
Dagsorden for ordinær generalforsamling
mandag den 9. maj 1988 kl. 18:30

N037
Datalogisk Institut
Universitetsparken 1
2100 København Ø.

1) Valg af dirigent.

2) Valg af referent.

3) Aflæggelse af beretning (ved formand)

4) Forelæggelse af revideret regnskab for 1987.

5) Fastsættelse af kontingent for 1989.

6) Valg af bestyrelse og to suppleanter.
Formand, et bestyrelsesmedlem samt suppleanter er på valg.

7) Valg af revisor samt suppleant.

8) Eventuelt.

- -

Efter ovenstående:

Z80 Brugergruppens Indkøbsforening
Dagsorden for ordinær generalforsamling

1) Valg af dirigent.

2) Valg af referent.

3) Aflæggelse af beretning af formand/næstformand.

4) Forelæggelse af revideret regnskab for 1987.

5) Forslag fra bestyrelsen om opløsning af Indkøbsforeningen.
(Hvis ikke 2/3 af medlemmerne er til stede og stemmer for opløsningen, indkaldes til en extraordinær generalforsamling indenfor 40 dage, hvor opløsning kan ske med 2/3 af de tilstedeværendes stemmer.)

5) Valg af bestyrelse samt 2 suppleanter.
(Skal evt. kun vælges til at afvikle indkøbsforeningen)

6) Valg af revisor samt suppleant til (rest af) 1988 regnskabet.

7) Evt.

MENU-PROGRAM til CP/M 2.2

Frank Damgaard

Med dette hjælpe-program kan det blive lettere at bruge CP/M, specielt for begyndere eller dem der ikke benytter CP/M særligt ofte.
For "garvede" brugere kan programmet også være en hjælp, idet "directory"-oversigten kun består af de programmer der kan udføres.

Når programmet kaldes, vises en oversigt over alle '.COM'-og '.OBJ' programmer. Desuden vises nogle enkelte CP/M kommandoer såsom:

TYPE REN LOAD SAVE DIR og ERA

I oversigten begynder '.OBJ'-programmer med et punktum "."

og CP/M kommandoer med en "*"

Med pil-tasterne flyttes det inverterede felt, så det ønskede program kan udvælges med ENTER/RETURN.

Før programmet kaldes, bedes man indtaste parametre til det valgte program, hvis der ikke er parametre til programmet tastes blot ENTER/RETURN.

EKS: Alle filer TEST.* ønskes slettet med ERA:

1) "Programmet ERA udvælges"
2) Argument to program call:
3) > TEST.*

Programmet er skrevet til COMPAS, men burde med kun mindre ændringer kunne bruges med PPAS, TURBO-PASCAL m.fl. Et krav er dog at den pågældende pascal-oversætter tillader BDOS-systemkald, samt kan skrive binære filer i "råt" format, f.eks. blockwrite i COMPAS og PPAS.

Programmet er oprindelig fra bladet MC, 1985, nr 10, s. 110 (Mikro Computer Zeitschrift, Franzis Verlag). Programmet er udvidet så den viste oversigt over programmer er alfabetisk sorteret.

Til dette benyttes den ofte anvendte quicksort-algoritme. Desuden er programmet udvidet så CP/M kommandoer og '.OBJ'-programmer også kan kaldes. Ændringerne benytter $$$.SUB filen for kommandoer og programmer der ikke er OBJ-filer.

I forbindelse med at $$$.SUB filen benyttes, skal man være opmærksom på at kald af en "submit"-fil med SUBMIT, ikke automatisk vil returnere til MENU-programmet. Skal der returneres til menu-programmet, må den sidste linie i submit-filen indeholde et kald af MENU.

-> -> ->

//

Leporelloliste:

Printerpapir i 'endeløse' baner, som er sammenfoldet, (zig zag). Papiret har huller i kanterne, til brug ved fremføring. (Leporello var tjener for Don Juan og skulle som sådan føre lange lister over sin herres damebekendtskaber...)

```
program CPM_2_2_Menu;
(* FROM: MC 10/1985 p110 (Franzis-verlag)                              *)
(* Changes made § 15 nov 1986 by Frank Damgaard                        *)
(* This program displays the '.COM' and '.OBJ' files of the            *)
(* disk as a menu. Furthermore the CP/M commands: LOAD, SAVE,          *)
(* REN, TYPE, DIR, and ERA are displayed.                              *)
(* Then the selected program is started.                               *)
(* The CCP may be patched to autostart this program. Thus              *)
(* CP/M becomes invisible to the user.                                 *)
(* The Menu-program does only work if it is compiled to a '.COM'-file *)
(* The Menu-program kan be aborted with the BREAK-character,           *)
(* usually control-C or control-D                                      *)
const
        (* The Cursor-keys must be changed to fit your CP/M-machine. *)
        (* Must be only one character *)
        UP=^K; DOWN=^V; RIGHT=^L; LEFT=^H; (* Moves cursor *)
        CR=^M; (* carriage-return / enter *)
        BREAK=^C; BREAKD=^D;
        TYPOBJ='OBJ'; TYPCOM='COM'; (* Program-types *)
        TRANSC=§0§0§0;  (* CP/M command-type *)
        maxcoms=132;    (* Maximum number of directory entries *)
        (* The menu-program must have the same name as "menu_name":*)
        menu_name='MENU';

type    FileDescp= record
          name: string(.8.);
          typ:  string(.3.);
        end;
        string126= string(.126.); (* string type for $$$.SUB file *)
        comindex= 0..maxcoms;

var     dma             : array(.0..127.) of byte;
        tabel_pt        : array(.comindex.) of comindex;
        com_files       : array(.comindex.) of FileDescp;
        com_file        : FileDescp;
        NoOfComs        : integer;
        ComPrLine,Lines : integer;
        selection       : integer;
        firstspace      : integer;
        ch              : char;
        progname        : string(.12.);
        progfile        : file;
        i               : integer;
        subfile         : FILE;
        exec_prog       : boolean;
        abort           : boolean;
        argument        : string126;

  procedure gotxy(x,y:integer);
  (* Denne procedure skal   ndres for andre pascal versioner!! *)
  begin
        gotoxy(x,y); (* compas 2.13dk *)
  end;

  procedure clear_screen;
  (* Denne procedure skal   ndres for andre pascal versioner!! *)
  begin
        gotoxy(0,0); clreos; (* compas 2.13dk *)
    (*  write(clrhom);       (* compas 3.03 , PPAS 3.10 *)
  end;
```

```
procedure highvideo;
(* Benyt sorte bogstaver med hvid baggrund *)
(* Denne procedure skal  ændres for andre pascal versioner!! *)
begin
      write(§27,'G4',^F); (* tvi950 terminal *)
end;

procedure lowvideo;
(* Benyt hvide bogstaver pa sort baggrund *)
(* Denne procedure skal  ændres for andre pascal versioner!! *)
begin
      write(§27,'G0'); (* tvi950 terminal *)
end;

procedure Scan_Directory(var entries: integer);
type  string11=string(.11.);
var   fcb             : array(.0..35.) of byte;
      a, i            : integer;
      counter         : integer;

  procedure storename(s1,s2:string11);
  (* Store direcory-name. *)
  begin
      com_files(.counter.).name:=copy(s1+'        ',1,8);
      com_files(.counter.).typ:=copy(s2+'   ',1,3);
      if counter<=maxcoms then counter:= succ(counter);
  end;

  procedure qsort(n:comindex);
  (* quicksort of command-array *)

    procedure sort(lf,rh:comindex);
    var i,j:comindex; x,w: comindex;
    begin
      i:= lf; j:= rh;
      x:= tabel_pt(.(lf+rh+1) shr 1.);
      repeat
        while com_files(.tabel_pt(.i.).).name<com_files(.x.).name do
              i:=succ(i);
        while com_files(.x.).name<com_files(.tabel_pt(.j.).).name do
              j:=pred(j);
        if i<=j then begin
          w:=tabel_pt(.i.); tabel_pt(.i.):=tabel_pt(.j.); tabel_pt(.j.):= w;
          i:=succ(i); j:=pred(j);
        end;
      until i>j;
      if lf<j then sort(lf,j);
      if i<rh then sort(i,rh);
    end;

  begin (* qsort *)
    sort(0,n);
  end;  (* qsort *)

  procedure Store_Entry(nof: integer);
  var filename: string11;
  begin
    nof:= nof*32 + 1;
    filename:= '';
    for i:=0 to 11 do
```

```
        filename:= filename+chr(dma(.nof+i.) mod 128);
      if (copy(filename, 9, 3) = TYPOBJ) or
         (copy(filename, 9, 3) = TYPCOM)
      then begin
         storename(copy(filename,1,8),copy(filename,9,3));
      end;
    end (* Store_Entry *);

  begin (* Scan_Directory *)
    counter:= 0;
    (* Set FCB to search for all files. ("????????.???") *)
    for i:=1 to 11 do fcb(.i.):= ord('?');
    fcb(.0.):= 0; (* current directory will be used *)
    fcb(.12.):=0; (* extent-number set to 00 *)
    bdos(26,addr(dma)); (* set dma-address to be to array "dma" *)
    (* call SEARCH-FOR-FIRST *)
    a:= bdosb(17,addr(fcb)); (* anderledes for TURBO *)
    while a<255 do begin
        Store_Entry(a);
        a:= bdosb(18,addr(fcb)); (* call SEARCH-FOR-NEXT *)
    end;
    (* Store CP/M commands : *)
    storename('LOAD',TRANSC);
    storename('SAVE',TRANSC);
    storename('REN',TRANSC);
    storename('TYPE',TRANSC);
    storename('DIR',TRANSC);
    storename('ERA',TRANSC);
    (* Initialize "tabel_pt" before sorting *)
    for i:=0 to maxcoms do tabel_pt(.i.):=i;
    if counter>2 then qsort(counter-1);
    entries:= counter;
  end (* Scan_Directory *);

  procedure Display_Prog_Name(n: byte);
  var   x, y:integer;
  begin
    x:= n div Lines * 12 + 1;
    y:= n mod Lines + 2;
    gotxy(x,y);
    com_file:= com_files(.tabel_pt(.n.).);
    if com_file.typ=TYPOBJ then write(' .')      (* for .OBJ-program *)
    else if com_file.typ=TRANSC then write(' *') (* for CP/M commands *)
    else write('  ');                            (* for .COM-program *)
    write(com_file.name,' ');
    gotxy(69,0);
  end;

procedure initsubmit;
  (* !!!!!! Denne del er for COMPAS. Proceduren skal højst sandsynlig *)
  (* være anderledes for andre  PASCAL-oversættere. !!!!!! *)
const fcbnavn=12; (* CP/M navn i FCB begynder 12 tegn efter fil-adresse *)
begin
  assign(subfile,'XXX.SUB');
  (* XXX erstattes med $$$ . Det er nødvendigt at gøre dette på
     denne måde, da COMPAS 2.13 kun tillader bogstaver og cifre
     som gyldige filnavne, (SUK!) *)
  for i:=1 to 3 do mem(.addr(subfile) + i + fcbnavn.):=ord('$');
  rewrite(subfile);
end;
```

```
procedure writesubmit(subn:string126);
  (* !!!!!! Denne del er for COMPAS. Proceduren skal muligvis *)
  (* være anderledes for andre  PASCAL-oversættere. !!!!!! *)
var     sector: array(.0..127.) of char;
        antal: integer;
begin
  sector(.0.):=chr(len(subn));
  for i:=1 to len(subn) do sector(.i.):=subn(.i.);
  sector(.len(subn)+1.):=§0;
  for i:=len(subn)+2 to 127 do sector(.i.):=^Z;
  blockwrite(subfile,sector,1); (* COMPAS 2.13; for TURBO: anderledes *)
(*blockwrite(subfile,sector,1,antal); (* PPAS 3.10 *)
end;

begin
  write(' Press any key to continue: '); read(kbd,ch);
  clear_screen;
  Scan_Directory(NoOfComs);
  (* compute then number of lines and columns to display the commands : *)
  ComPrLine:=trunc(sqrt(NoOfComs)+0.001);
  Lines:= ComPrLine;
  if ComPrLine>6 then ComPrLine:=6;
  while  Lines * ComPrLine < NoOfComs do Lines:=succ(Lines);
  writeln('<SPACE> or <ARROWS> to advance, <RET> to select program.');
  lowvideo;
  for i:=0 to pred(NoOfComs) do Display_Prog_Name(i);
  highvideo;
  Display_Prog_Name(0);
  lowvideo;
  selection:=0;
  abort:=false; exec_prog:=false;
  repeat
    read(kbd,ch);
    case ch of
     RIGHT: begin (* Cursor right *)
        lowvideo;
        Display_Prog_Name(selection);
        selection:= selection+Lines;
        if selection>=NoOfComs then selection:=selection mod Lines;
        highvideo;
        Display_Prog_name(selection);
      end;
     LEFT: begin (* Cursor left *)
        lowvideo;
        Display_Prog_Name(selection);
        selection:= selection-Lines;
        if selection<0 then selection:=selection + Lines*ComPrLine;
        if selection>=NoOfComs then selection:=selection-Lines;
        highvideo;
        Display_Prog_name(selection);
      end;
     DOWN: begin (* Cursor down *)
        lowvideo;
        Display_Prog_Name(selection);
        if selection mod Lines = Lines-1 then selection:=selection-Lines+1
        else selection:=selection+1;
        if selection>=NoOfComs then selection:=(ComPrLine-1)*Lines;
        highvideo;
        Display_Prog_name(selection);
      end;
```

```
    ' ': begin (* Next program-name *)
       lowvideo;
       Display_Prog_Name(selection);
       selection:=succ(selection);
       if selection >= NoOfComs then selection:=0;
       highvideo;
       Display_Prog_name(selection);
     end;
    UP: begin (* Cursor up *)
       lowvideo;
       Display_Prog_Name(selection);
       if selection mod Lines = 0 then selection:=selection+Lines-1
       else selection:=pred(selection);
       if selection>=NoOfComs then selection:=NoOfComs-1;
       highvideo;
       Display_Prog_name(selection);
     end;
    BREAK,BREAKD: abort:=true; (* abort menu-program *)
    CR: exec_prog:=true;       (* call/enter selected program *)
    otherwise                  (* TURBO:  else *)
   end;
   lowvideo;
 until abort or exec_prog;
 if exec_prog then begin
   com_file:= com_files(.tabel_pt(.selection.).);
   progname:=com_file.name;
   clear_screen; highvideo;
   write('  ',progname,' '); lowvideo;
   writeln; writeln(' Argument to program call:'); write('>');
   read(argument);
   (* delete trailing spaces from "progname" : *)
   firstspace:= pos(' ',progname);
   if firstspace > 0 then
     delete(progname,firstspace,len(progname)-firstspace+1);
   clear_screen;
   initsubmit; (* make $$$.SUB - file *)
   (* program-calls are added in reverse order to $$$.sub *)
   writesubmit(menu_name); (* last: call MENU program *)
   if com_file.typ=TYPOBJ then begin
     (* PASCAL .OBJ-program *)
     close(subfile);
     (* Execute pascal program directly by CHAIN-command *)
     assign(progfile,progname+'.'+com_file.typ);
     chain(progfile);
   end else begin
     (* CP/M-command or .COM-program *)
     if LEN(argument)>0 THEN argument:=copy(' '+argument,1,117);
     writesubmit(progname+argument); (* first: call selected program *)
     close(subfile);
   end;
 end;
 if abort then begin
       clear_screen; write('-- ',menu_name,' ABORTED -- ');
 end;
end.
```

- - - - -

NB. § tegnet skal erstattes med masterspace (= pound sign) (= snabel-a)

Salg / Indkøbsforening / Brugergruppe

20	stk.	8"	3M-743 disketter DSDD	a	30 kr.
5	-	3.5"	3M-743 disketter SSDD	a	20 kr.
3	-		Head Cleaning Set 5.25"	a	50 kr.
1	-		Pascal-bog		140 kr.
		'En grundlæggende indføring i Pascal'			
		(Per Amdal Steffensen & Leif Pehrsson / Forlaget Systime)			
2	stk.	'Maskinkodeprogrammering med Z80' (Jesper Skavin)		a	120 kr.
14	-		JRT-Pascal Manual (I + II)	a	75 kr.
3	-		CP/M Mappe	a	100 kr.

(priser incl. moms., excl. porto + forsendelse)

ANNONCER

NASCOM-2 med original strømforsyning, floppy disk controler kort, G802 Ram-kort med 64K, motherboard med 4 slots, Eprom brænder, netstøjfilter indbygget i kabinet med tastatur.

5.25" floppydisk indbygget i MK kabinet med strømforsyning.

Printer: STAR DP 8480 matrixprinter med seriel interface.

Ring og giv et bud. Evt. bytte med brugt, velfungerende VHS video.

Lars-Jørn Christiansen. Møllebakken 11, 9240 Nibe

Telf. (08) 35 24 06, arb. (08) 15 62 22 lokal 2633

- - -

NASCOM-2 bestående af Nascom-2 CPU-kort, Gemini RAM-kort med 64k, Gemini FDC kort samt Gemini IVC-kort. - Det hele i en pæn og meget solid sort/grå metalkasse med mål ca. 38,5 x 21,5 x 11 cm.

En 80 Watt strømforsyning i tilsv. metalkasse, ca. 23,5x21,5x11 cm. Metalkabinet som ovenfor til Nascom-2, beregn. til 2 stk. diskdrev.

1 Eprombrænder til 3 forsk. typer EPROM, med program i båndversion samt manual. Alt fra Poly-Data, beregnet for Nas.Sys eller andre, der har to 8-bit parallelporte.

Fuld dokumentation og diagrammer medfølger til alt.

Det hele sælges for højeste bud.

Misser Jensen. Grønningen 56, 4220 Korsør. Telf. (03) 57 16 22

'ALTERNATIV' GRAFiK

Ebbe Sønderhousen

Efter læsning af John B. Jacobsens udmærkede artikel om, hvordan en linie og en cirkel tegnes, kom jeg i tanke om at jeg for flere år siden selv havde siddet og rodet med noget..... Frem med de gamle støvede disketter og igang.

Jo, ganske rigtigt.

Hvorfor jeg overhovedet blander mig i John B.'s artikel er at jeg lavede rutinerne i Assembler UDEN BRUG AF GANGE OG DIVISION. Jeg har ikke prøvet, men jeg vil vove den påstand at den vil gå langt hurtigere.

Nu er jeg i den situation, at jeg ikke længere er i besiddelse af en maskine med grafik, så jeg har skrevet rutinen om, så den 'tegner' stjerner på skærmen ved brug af cursor-adressering. Det kan vel være lige så godt til testbrug.

Nå, men til sagen:

Fremgangsmåden er som følger: En linie kan betragtes som et startpunkt og en hældning. Med hældning mener jeg at linien går et antal (dx) positioner udad X-aksen og et antal (dy) udad Y-aksen. Disse fire parametre (startpunkt (x,y) og hældning (dx,dy)) er kravet til rutinen for at kunne tegne en linie.

Fidusen er så at der defineres to tælleværker tvx og tvy hvori hhv. dx og dy adderes igen og igen indtil tvx eller/og tvy giver carry. Hvis tvx giver carry 'vandres' een position udad X-aksen, og der sættes en dot/stjerne. Ved carry på tvy sættes dot/stjerne een position længere ude ad Y-aksen.

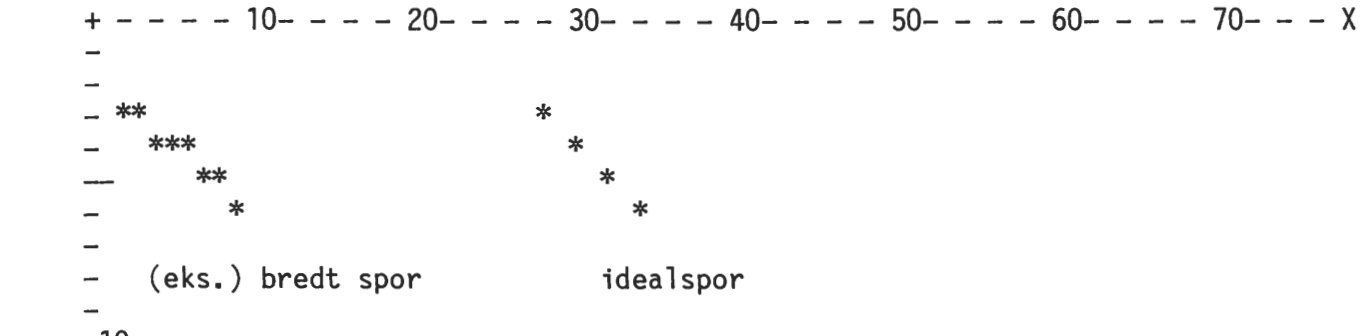
Nu vil det kvikke hoved jo så sige: "Ja-men, hvis nu (dx,dy) = (7,13), så skal der adderes 9362 gange for hver dot i X-aksens retning og ca. det halve i Y-aksens retning". Det er også rigtigt. Hvis vi ikke lavede lidt hokus pokus inden vi startede adderings loopet. Vi skal nemlig ikke betragte dx og dy, men forholdet mellem dem. Det vil medføre at forholdet (dx,dy) er det samme som (2*dx,2*dy). Liniedannelsen skal så på anden vis 'bremses' når den når (x+dx,y+dy). Jeg definerer derfor felterne hx og hy, som indeholder hældningen. Ved at shifte hx og hy til venstre (ganger med 2) indtil een af dem får en 1-bit i mest betydende bit bibeholder vi forholdet og opnår at vi næsten hver gang vi adderer opnår carry i enten tvx eller tvy.

Der kan bremses til rette tid ved at nedtælle de oprindelige dx og dy, hver gang vi rykker udad hhv. X-aksen og Y-aksen.

Lad os se på et eksempel.

Vi skal tegne en linie startende i (2,3) med hældning (7,3).

X	Y	dX	dY	hX	hY	tvX	tvY		
0002	0003	0007	0003						
				E000	6000			:	hældning udfra (dx,dY)
						E000	6000	:	1. add ingen carry
0003		0006				C000	C000	:	2. add carry X
0004	0004	0005	0002			A000	2000	:	3. add carry X og Y
0005		0004				8000	8000	:	4. add carry X
0006		0003				6000	E000	:	5. add carry X
0007	0005	0002	0001			4000	4000	:	6. add carry X og Y
0008		0001				2000	A000	:	7. add carry X
0009	0006	0000	0000			0000	0000	:	8. add carry X og Y



Nedenfor er vist et program, der udnytter ovenstående algoritme.

Følgende kommentarer knytter sig til algoritme og program:

Der tages ikke højde for fortegn, dvs. at vi kun arbejder med startpunkter i een kvadrant og med hældninger, der peger længere ud i samme kvadrant, det er enkelt at gøre, men jeg har undladt det, for at undgå at det vigtige fortaber sig i tågerne.

Det spor linien følger kan afvige +/- 1 fra det ideelle spor. Det vil under de fleste omstændigheder ikke have nogen betydning. Derimod er det af større betydning at der ofte vil fremkomme et bredt spor (se nedenfor), hvilket jeg i en senere artikel vil komme nærmere ind på.

```
.comment *---------------------------------------+
/                                                /
/   LINIR.MAC - EBBE SØNDERHOUSEN - Z80-NYT      /
/   - 403 -     1988-02-14. ( X+Y skala ).       /
/               M80 / L80                        /
/                                                /
*-----------------------------------------------+

        .Z80
        ASEG
        ORG     0100H

;------ Kontroltegn --------------------

NUL     EQU     00H
LF      EQU     0AH
CR      EQU     0DH
ESC     EQU     1BH
XX      EQU     20H     ; offset skærm

;------ CP/M adresser ------------------

WBOOT   EQU     0000H
BDOS    EQU     0005H

;------ BDOS rutiner -------------------

B$CONOUT        EQU     2
B$READCONS      EQU     10
```

```
LINE:   CALL    GETPARM         ; Spørg efter X,Y,DX,DY
        CALL    DRAW            ; Tegn linien

        CALL    CPARK           ; Cursor parkering
        JP      WBOOT           ; Retur til CCP
;=======================================================

X:      DS      2
Y:      DS      2
DX:     DS      2
DY:     DS      2

;--- Modtag informationer om liniens startpunkt og hældning ---

GETPARM:
        CALL    PROMPT
        DB      CR,LF,'Tast startpunkts X-koordinat: ',NUL
        LD      (X),HL
        CALL    PROMPT
        DB      CR,LF,'Tast startpunkts Y-koordinat: ',NUL
        LD      (Y),HL
        CALL    PROMPT
        DB      CR,LF,'Tast hældning i  X-retningen: ',NUL
        LD      (DX),HL
        CALL    PROMPT
        DB      CR,LF,'Tast hældning i  Y-retningen: ',NUL
        LD      (DY),HL
        CALL    CLEAR
        CALL    VSKALA          ; + lodret skala
        RET

;--- Vis meddelelsen stacktop peger på og afvent indtastning ---
;       af et decimaltal, som returneres i HL.

PROMPT: EX      (SP),HL         ; textpointer (stacktop) til HL.
        CALL    OUTSTR
        EX      (SP),HL         ; returadresse (1. byte efter text)
                                ; til stacktop.
        CALL    INSTR
        DW      PROMPTINPUT
        CALL    ATOI
        DW      PROMPTINPUT+1
        RET

PROMPTINPUT:
        DB      6               ; Længde på text buffer
        DS      1               ; Længde af indtastet text
        DS      6               ; Text buffer

;--- Vis den NUL-afsluttede meddelelse, som HL peger på ---

OUTSTR: LD      A,(HL)          ; Næste tegn i text
        INC     HL              ; Text pointer frem
        OR      A               ; Er tegn NUL
        RET     Z               ; Hvis ja: slut
        PUSH    HL              ; Gem text pointer
        CALL    OUTCHR          ; Vis tegn
        POP     HL              ; Hent text pointer
        JR      OUTSTR          ; På'n igen
```

```
;------ Vis tegn indeholdt i A --------

OUTCHR:
        LD      E,A             ; Flyt tegn på plads til BDOS
        LD      C,B$CONOUT      ; Funktionskode = fu2
        CALL    BDOS            ; Vis tegn
        RET

;------ Afvent indtastning af tegn i buffer udpeget af stacktop ------

INSTR:  EX      (SP),HL         ; Adr. på buffer hentes fra stacktop
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        INC     HL
        EX      (SP),HL         ; Returadr. lægges tilbage på stacktop
        LD      C,B$READCONS    ; Funktionskode
        CALL    BDOS            ; Indtast text i text buffer
        RET

;-------- Konverter indtastet decimaltal til dets hex. værdi ---------
;         og returner resultatet i HL.

ATOI:   EX      (SP),HL         ; Adr. på text hentes fra stacktop
        LD      E,(HL)
        INC     HL
        LD      D,(HL)
        INC     HL
        EX      (SP),HL         ; Returadr. lægges tilbage på stacktop
        LD      HL,0            ; Resultat initieres med 0 (nul)
        LD      A,(DE)          ; Længden af text hentes
        OR      A               ; Er længden af text 0?
        RET     Z               ; Hvis ja: slut
        LD      B,A             ; Overfør textlængde til tæller
ATOI1:  PUSH    DE              ; Gem text pointer
        ADD     HL,HL           ; Gang resultat med 10: her * 2
        PUSH    HL              ; Gem resultat * 2
        ADD     HL,HL           ; * 4
        ADD     HL,HL           ; * 8
        POP     DE              ; Hent resultat * 2
        ADD     HL,DE           ; res. * 2 + res. * 8 = res. * 10
        POP     DE              ; Hent text pointer
        INC     DE              ; Pointer frem
        LD      A,(DE)          ; Hent ciffer
        PUSH    DE              ; Gem text pointer
        SUB     '0'             ; Tegn --> værdi
        LD      E,A             ; værdi lægges til resultat
        LD      D,0
        ADD     HL,DE
        POP     DE              ; Hent text pointer
        DJNZ    ATOI1           ; På'n igen, hvis flere cifre
        RET
;
;------ Tegn linie givet ved startpunkt (x,y) med hældning (dx,dy) ---
;
DRAW:   CALL    OUTDOT          ; Tænd dot i startpunkt
        LD      HL,(DX)
        LD      (RX),HL         ; Hældning overføres til rest
        LD      (HX),HL         ;    -        -      - hældning
```

```
        LD      HL,(DY)
        LD      (RY),HL         ; Hældning overføres til rest
        LD      (HY),HL         ;    -         -      - hældning
        CALL    OPTIMERHELD
DRAW.X: LD      HL,(TVX)        ; Adder hældning til tælleværk
        LD      DE,(HX)
        ADD     HL,DE
        LD      (TVX),HL
        JR      NC,DRAW.Y       ; Hvis addition giver CARRY
        LD      HL,(X)          ;   flyttes 1 pos i x-retningen
        INC     HL
        LD      (X),HL
        LD      HL,(RX)         ; Rest tælles ned
        DEC     HL
        LD      (RX),HL
DRAW.Y: LD      HL,(TVY)        ; adder hældning til tælleværk
        LD      DE,(HY)
        ADD     HL,DE
        LD      (TVY),HL
        JR      NC,DRAW.DOT     ; Hvis addition giver CARRY
        LD      HL,(Y)          ;   flyttes 1 pos i y-retningen
        INC     HL
        LD      (Y),HL
        LD      HL,(RY)         ; Rest tælles ned
        DEC     HL
        LD      (RY),HL
DRAW.DOT:
        CALL    OUTDOT
        LD      HL,(RX)         ; Hvis rest er 0 for både x og y: slut
        LD      A,H
        OR      L
        JR      NZ,DRAW.X
        LD      HL,(RY)
        LD      A,H
        OR      L
        JR      NZ,DRAW.X
        RET
TVX:    DS      2
TVY:    DS      2
RX:     DS      2
RY:     DS      2
HX:     DS      2
HY:     DS      2

;------ Tænd en dot i punktet (x,y) ********** MASKINAFHÆNGIG ------

OUTDOT: LD      A,(X)
        ADD     A,' '
        LD      (CURSOR.KOL),A
        LD      A,(Y)
        ADD     A,' '
        LD      (CURSOR.LIN),A
        LD      HL,CURSOR
        CALL    OUTSTR
        LD      A,'*'
        CALL    OUTCHR
        RET
CURSOR:         DB      ESC,'Y'   ; lead-in
CURSOR.LIN:     DS      1
CURSOR.KOL:     DS      1
                DB      NUL
```

```
;------ HX og HY shiftes til venstre indtil en af dem giver CARRY ---

OPTIMERHELD:
        LD      HL,(HX)         ; shift left ved brug af ADD
        ADD     HL,HL
        RET     C               ; retur hvis shift giver CARRY
        EX      DE,HL           ; gem shiftet resultat
        LD      HL,(HY)
        ADD     HL,HL
        RET     C               ; retur hvis shift giver CARRY
        LD      (HY),HL         ; shiftede resultater får kun virkning,
        LD      (HX),DE         ;   hvis CARRY er 0 efter begge shifts
        JR      OPTIMERHELD

;------ Slet skærm og cursor HOME ********** MASKINAFHÆNGIG ---

CLEAR:  LD      HL,CLS          ; CLEAR: LD     A,1AH
        CALL    LAES            ;        CALL   OUTCHR
        RET
   ;--------------------

LAES:   LD      A,(HL)
        CP      07EH            ; ~ (Tilde /el.tilsv./ = slut på text)
        RET     Z
        PUSH    HL
        CALL    SKRIV
        POP     HL
        INC     HL
        JR      LAES
        RET
   ;--------------------

SKRIV:  LD      C,02            ; fu2
        LD      E,A
        CALL    BDOS
        RET
;----------------------------------------

CLS:    DB      ESC,'H',ESC,'J'  ; HOME, CLS, derefter X-skala

XAKS:   DB      '+ - - - - 10- - - - 20- - - - 30- - - - '
        DB      '40- - - - 50- - - - 60- - - - 70- - - - 8~'

VSKALA: LD      C,9             ; fu9
        LD      DE,YAKS         ; Y-skala
        CALL    BDOS
        RET
YAKS:   DB      ESC,'Y',1+XX,XX,'-',CR,LF,'-',CR,LF,'-',CR,LF,'-'
        DB      CR,LF,'--',CR,LF,'-',CR,LF,'-',CR,LF,'-',CR,LF,'-'
        DB      CR,LF,'-10',CR,LF,'-',CR,LF,'-',CR,LF,'-',CR,LF,'-'
        DB      CR,LF,'--',CR,LF,'-',CR,LF,'-',CR,LF,'-',CR,LF,'-'
        DB      CR,LF,'-20',CR,LF,'-',CR,LF,'-',CR,LF,'-$'
;----------------------------------------
CPARK:  LD      C,9      ; fu9
        LD      DE,SIDST ; anbring cursor udenfor feltet
        CALL    BDOS
        RET
SIDST:  DB      ESC,'Y',24+XX,XX,'$'
;----------------------------------------
        END     LINE
```

C++

Objektorienteret programmering i C

Bjarne Gertz Pedersen

C++ er en overbygning til det meget anvendte sprog C. Den væsentligste forbedring er, at der i C++ gives fleksible og effektive muligheder til at definere nye typer. Det grundlæggende koncept i C++ er således klassebegrebet, kendt fra Simula67 og Concurrent Pascal. Klassen er en brugerdefineret type, som giver mulighed for at knytte datastrukturer og operationer sammen.

Denne teknik kaldes for data abstraktion. Det fundamentale i abstrakte datatyper er at skille implementationen fra anvendelsen.

Da kun operationerne er tilgængelige, opnås stor sikkerhed og implementeringsdetaljerne kan skjules. Programmet kan derved deles ned i en række enheder, der ligger tæt på den opgave, som skal løses.

Baggrund

C++ er udviklet på Bell Labs., hvor det almindelige C sprog også har sin oprindelse. Der har ikke været tale om et egentligt C++ projekt, men ideen til sproget opstod, da Bjarne Stroustrup skulle skrive nogle hændelsesstyrede simuleringsprogrammer.

Til denne opgave ville Simula67 have været ideel, men pga. effektivitetshensyn blev det ikke valgt. I 1983 blev C++ for første gang installeret uden for Bjarne Stroustrups udviklingsgruppe på Bell Labs. og i dag er sproget det mest anvendte objekt orienterede sprog i USA.

Kompatibilitet

Et væsentligt designkriterium for C++ har været at bibeholde kompatibiliteten med C. Den basale syntaks og semantik er ens for de to sprog.

Dette betyder at:

- de biblioteksrutiner, der er skrevet i C, også kan bruges i C++ programmer. C og C++ kan lænkes sammen.

- de mange C programmører umiddelbart kan anvende C++ og når det bliver nødvendigt med bedre muligheder for typecheck og data abstraktion, kan disse faciliteter tages i brug.

C++ "oversætteren" er ikke en almindelig oversætter i den forstand, at den genererer objektkode, men derimod en translator, der virker som front-end til en almindelig C oversætter, dvs. den genererer C kode, som derefter skal oversættes af en almindelig C oversætter.

Der findes C++ translatorer til en lang række C oversættere og herunder kan jeg blot nævne nogle af de mere kendte:

DOS: - Lattice, Microsoft.

Andre: - VAX C, SUN C, APOLLO, HP-9000, XENIX, UNISOFT.

Desuden findes det kendte PC bibliotek PforCe nu også i en PforCe++ version.

Effektivitet

C++ bibeholder C's effektive muligheder for at kommunikere med hardware. Da de indførte begreber i C++ kun giver translatoren bedre muligheder for kontrol i oversættelsesfasen og da den genererede C kode oversættes af almindelige C-oversættere, vil der ikke gå noget tabt mht. eksekveringshastighed, men derimod vil der kunne vindes meget i udviklingstid, især udvikling af større program-komplekser.

C++ udvidelser

Som allerede antydet er en række forbedringer indført i forhold til standard C. Den væsentligste er muligheden for at anvende brugerdefinerede typer. Klassen indeholder en privat datastruktur, samt en mængde operationer, der kan operere på denne datastruktur.
Et eksempel på en klasse kan være "ostream" fra I/O biblioteket. Den indeholder en buffer, hvor tegn til standard output kan gemmes, samt de operationer, der kan gemme tegnene i bufferen. Erklæringen ser således ud:

```
class    ostream{
         streambuf * buf;
         int state;
public   void put (char*);
         void put (long);
         void put (double);
};
```

buf og state er de private datastrukturer, mens put operationerne er interfacen for programmer, der ønsker at skrive til standard output. En erklæring af en klasse beskriver blot typen af klassen og derfor er det nødvendigt at definere en variabel af klassen. I det følgende vil en variabel blive benævnt som et objekt. Helt præcist, i følge terminologien inden for objekt orienteret programmering, er det kun datastrukturerne, der er objekter, mens operationerne er meddelelser, som sendes til et objekt af en klasse.

C++ notationen understøtter ikke helt tankegangen i objekt orienteret programmering, hvor der altid sendes meddelelser til objekter, så derfor vil jeg i det følgende ikke anvende ordet meddelelse for en operation eller funktion, men dog bibeholde benævnelsen objekt for en variabel af en datastruktur.

I erklæringen af klassen fremgår det, at der er 3 operationer med samme navn, put(), men med forskellige parameter overførsler. Det er tilladt i C++ at anvende samme navn for forskellige operationer. Dette er specielt anvendeligt, når man har en række operationer, der udfører næsten den samme ting, men som her på forskellige datatyper. Dette kaldes for "overloaded function names" og oversætteren skelner mellem funktionerne på baggrund af parameter overførslen.

Hierarki af klasser

Et af de bedre redskaber til at organisere komplekse problemstillinger er at anvende hierarkiske strukturer. Det gælder om at anvende en træstruktur, hvor objektet i roden indeholder de mest generelle ting og jo længere man kommer ud af forgreningerne jo mere specialiseret bliver objekterne.

Denne form for opbygning er en af de fundamentale ting, der skal understøttes af objekt orienterede sprog. I C++ gøres det ved at klasser kan "arve" hinandens egenskaber, på engelsk benævnes det som "Inheritance".

Hvis man skulle beskrive eller organisere musikinstrumenter kunne det gøres ved at have en basisklasse "Baseclass", hvor de generelle egenskaber, der gælder for alle musikinstrumenter er defineret.

Forgreningerne kunne derefter være strengeinstrumenter, blæsere, slagtøjsinstrumenter m.m. De klasser vil, foruden deres egne specielle egenskaber, stadig også være omfattet af basisklassens egenskaber.

Disse klasser kaldes afledte klasser "Derived Classes". Klassen af strengeinstrumenter ville så igen kunne opdeles i en række afledte klasser, nemlig strygere, guitarer m.m. og på denne måde kunne der fortsættes.

Basisklassen for strygere ville nu være strengeinstrumenter.

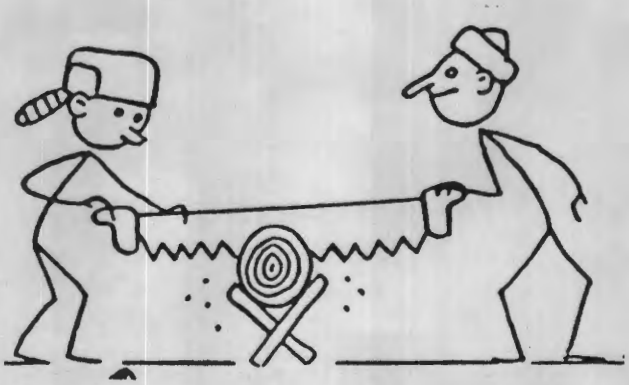
Hvis dette føres videre til C++ betyder det, at hvis et objekt af en klasse bedes om at udføre en operation vil den også kunne bruge de operationer, der er defineret i dens basisklasse, hvis det er nødvendigt for at udføre operationen.

Dette betyder igen, at afledte klasser kun bør tilføje operationer eller egenskaber til hierarkiet.

Der findes langt flere begreber i C++, men det vil være for omfattende at komme nærmere ind på disse i denne artikel. Jeg kan blot nævne nogle af dem, f.eks. virtuelle funktioner, garanteret initialisering af klassedata, dynamisk typebinding, inline substituering af funktioner, operatorer til lagerstyring, operator overload og default parametre.

- - - -

Reprint af artikel skrevet i forb. med Teknologisk Institut / Automatiseringsteknik's kursus 6-8 April & 6-8 September 1988. Yderligere oplysninger om C++ og kursus kan fås ved henv. til Bjarne Gertz Pedersen, (02) 99 66 11 lokal 142



Cee - Saw

Hexadecimal MULTIPLIKATIONS-TABEL

1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	04	06	08	0A	0C	0E	10	12	14	16	18	1A	1C	1E
3	06	09	0C	0F	12	15	18	1B	1E	21	24	27	2A	2D
4	08	0C	10	14	18	1C	20	24	28	2C	30	34	38	3C
5	0A	0F	14	19	1E	23	28	2D	32	37	3C	41	46	4B
6	0C	12	18	1E	24	2A	30	36	3C	42	48	4E	54	5A
7	0E	15	1C	23	2A	31	38	3F	46	4D	54	5B	62	69
8	10	18	20	28	30	38	40	48	50	58	60	68	70	78
9	12	1B	24	2D	36	3F	48	51	5A	63	6C	75	7E	87
A	14	1E	28	32	3C	46	50	5A	64	6E	78	82	8C	96
B	16	21	2C	37	42	4D	58	63	6E	79	84	8F	9A	A5
C	18	24	30	3C	48	54	60	6C	78	84	90	9C	A8	B4
D	1A	27	34	41	4E	5B	68	75	82	8F	9C	A9	B6	C3
E	1C	2A	38	46	54	62	70	7E	8C	9A	A8	B6	C4	D2
F	1E	2D	3C	4B	5A	69	78	87	96	A5	B4	C3	D2	E1

////////////////////////////// PROGRAM-SERVICE //////////////////////////////

Ved henvendelse til program-forfatterne / redaktionen vil det fremover være muligt, for en mindre (symbolsk?) betaling, at få programmer fra bladet overført til/på ens egen, tilsendte porto-fri osv. diskette, så spares indtastningen Der rådes over en række forskellige formater og i 'svære' tilfælde kan vi jo altid falde tilbage på mindst 8 forsk. IBM-formater, heriblandt, næsten alle 'PC'-typerne.