Indledning.

Spillet som denne rapport beskriver, indgår i et større program, der er lavet som projekt i valgfaget programmering C på HTX i perioden 9/11-98 til 12/1-99.

Spillet skal give de forskellige deltagere i projektet en ide om hvordan man håndterer større programmer og koordinerer indsatsen omkring et projekt, der drejer sig om programmering. Det er også meningen at de forskellige deltagere skal blive mere øvede i at dokumentere programmer.

Indholdsfortegnelse:

Indledning. 1
Indholdsfortegnelse: 1
Problemformulering. 2
Ideen med spillet. 2
Brugervejledning. 2
Skærmlayout. 3
Dokumentation for selve koden. 5
NS-diagrammer for Master mind. 5
Interface til Master Mind delen af programmet. 7
Konstanter brugt i programmet. 7
Variabler brugt i programmet. 7
Interne procedurer og funktioner brugt i programmet. 8
Konklusion. 8
Bilag. 9
Bilag 1. Selve koden. 9
Bilag 2. Test Program, brugt til at afprøve mm_unit med. 16
Bilag 3. Test-unit der var nødvendig for at afprøve programmet. 16

Problemformulering.

Opgaven går ud på at designe et spil, der kan indgå i en større sammenhæng af spil.

- Hvilket spil skal det dreje sig om?
- Hvilket udseende skal spillet præsenteres med?
- Hvordan skal brugerinterfacet ellers være (indtastninger med videre)?
- Hvilke regler skal spillet følge?
- Hvordan bliver det integreret i en større sammenhæng med andre spil?

Ideen med spillet.

Ideen i spillet er, ligesom i det rigtige master mind, af man skal gætte en skjult kode.

Koden består af f.x. 4 brikker med hver sin farve.

Spillet kan gøres lettere eller sværere alt efter hvor mange brikker man skal gætte, hvor mange forskellige farver de må have, og om der må være flere af same farve.

Spillet skal opbygges, så det kan arbejde på en DOS tekst-skærm. hvis der er problemer med at have det hele, kan skærmen udvides ved at arbejde med en 43-liniers skærm.

Brugervejledning.

Først gives en brugervejledning, der indeholder hvad spillet går ud på, og at man i spillet kan trykke F1 for hjælp.

Først skal sværhedsgraden vælges. Der er 8 forskellige sværhedsgrader.

1. 3 brikker, 4 farver, ingen ens.
2. 3 brikker, 4 farver. med ens
3. 3 brikker, 6 farver ingen ens.
4. 3 brikker, 6 farver med ens.
5. 4 brikker, 5 farver, ingen ens.
6. 4 brikker, 5 farver. med ens
7. 5 brikker, 7 farver ingen ens.
8. 5 brikker, 7 farver med ens.

Når man har valgt sværhedsgraden slettes skærmen og man får første forslag til kombination op på skærmen. Den kan man rette i ved at bladre hen over brikkerne med højre-/venstre-pil, og man kan skifte farven med op-/ned-pil. Når man er tilfreds med sit gæt taster man ENTER. Så angives på skærme hor mange rigtigt placerede og hvor mange derudover der var af rigtige farver, men forkert placerede. Herefter starter man med næste gæt, med den kombination man gættede sidst.

ESC bringer en ud af spillet, og tilbage i hovedmenuen. Man får muligheden for at fortryde , så man komme tilbage i spillet hvis man ikke taster Enter.

F1 som sagt hjælp.

F9 kan i første omgang bruges som snydetast, til at få den rigtige kombination frem på skærmen, dette kan være godt til test – den skal i den rigtige version luses ud.

Alle andre taster skal give en beep.

Når man gætter rigtigt får man en tillykkeskærm, og man skal taste enter, herefter kommer man tilbage i hovedmenuen.

Hvis man ikke når at gætte rigtigt inden skærmen er fuld, får man en BUUH-skærm og man skal taste Enter.

Efter endt spil kommer man tilbage til hovedmenuen (vælg hvilket spil).

Skærmlayout.

Indledningsskærm:

```
                        M A S T E R   M I N D

Velkommen til Master Mind
Du skal gætte en kombination af farvede brikker.
Ved hvert gæt får du at vide hvor mange brikker der er rigtig farve og
rigtigt placeret, og du får at vide hvor mange derudover, der har den
rigtige farve, men er forkert placeret.

Ønsker hjælp til hvordan spillet spilles, trykkes F1 i spillet.

Spillet kan gennemføres i forskellige sværhedsgrader, som følger:
1.   3 brikker  4 farver    uden ens
2.   3 brikker  4 farver    med ens
3.   3 brikker  6 farver    uden ens
4.   3 brikker  6 farver    med ens
.
.
.
.
```

Hjælpeskærm:

```
                        M A S T E R   M I N D
                              H J Æ L P

←     vælger brikken til venstre for (hvis der er flere).
→     vælger brikken til højre for (hvis der er flere).
↑     vælger farven før den valgte.
↓     vælger farven efter den valgte.
Esc   afslutter spillet i utide
F1    giver denne hjælpeskærm
Enter godkender gættet

Tast Enter for at komme tilbage til spillet.
```

Spil Skærm:

```
                          M A S T E R    M I N D

1. Gæt : □ □ □ □   2 korrekte   1 farve

2. Gæt : □ □ □ □   1 korrekt    2 farver

3. Gæt : □ □ □ □

.

.
```

Tillykke Skærm:

```
                          M A S T E R    M I N D

Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke
Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke
Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke

Du nåede det på 4 gæt

Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke
Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke
Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke Tillykke

Tast ENTER
```

Sorry Skærm:

```
                          M A S T E R    M I N D

Sorry din tumpe, du kunne ikke nå det inden for 12
gæt, så nu slutter festen.

Tast ENTER
```

Afbrydnings Skærm (i bunden af skærmen):

```
Du afslutter nu spillet uden at fuldføre det.  Tast ENTER
```

Dokumentation for selve koden.

NS-diagrammer for Master mind.

Disse NS-diagrammer beskriver ikke fuldt hvad der sker i hver enkelt linie, men giver en beskrivelse, der kan skabe overblik over hvordan mm fungerer.

Proceduren mm er hovedrutinen, der kaldes fra hovedmenuen i programmet, den er rammen om master mind spillet.

Skriv en startskærm
Indlæs sværhedsgraden
Indstil parametrene efter hvilken sværhedsgrad der er valgt
Generer det skjulte gæt og start med ens farver
Skriv op hvilket gæt det er
Skriv hvordan gættet ser ud
Reager på indtastning, og ret på gættet ud fra det
Repetér indtil der tastes ENTER
Test på om gættet er rigtigt, og skriv hvor mange der er korrekte
Gem gættet i gamle gæt ud fra hvilket gæt det er (bruges til at reetablere efter hjælp).
Repetér indtil spillet er færdigt eller bliver afbrudt

Function test_rigtig kaldes, når brugeren har fået stillet sit gæt op, og rutinen tester først om hele gættet er korrekt, og derefter hvor mange der er rigtige, både helt rigtige og rigtig farve, men forkert placeret. Funktionen skriver hvor mange der er af hver.

Sæt returværdien til sand	
Sæt tælleværdierne for rigtige og rigtig farve, men forkert placeret til nul	
Tæl alle brikkerne igennem	
Marker at brikken ikke er talt, hverken i gættet eller i den rigtige	
Er gættet som den rigtige kode for brikken?	
Nej	Ja
Sæt returværdien til at være falsk	Marker at brikken er talt, som rigtig og gæt
	Tæl antal rigtig i gættet op
Tæl alle brikkerne igennem med N som index	
Er brik N talt i den rigtige kode?	
Ja	Nej
Tæl alle brikkerne igennem med I som index	
Er brik I talt i gættet?	
Ja	Nej
Er farven for rigtig index N = farven for gæt index I?	
Nej	Ja
	Marker brikken med index I er talt i gættet
	Marker brikken med index N er talt i de rigtige
	Tæl rigtig farve, men forkert placeret op
	Afslut tællingen med index I
Skriv hor mange rigtige der er (rigtig farve på den rigtige placering)	
Skriv hvor mange der er af rigtig farve (en rigtig farve, men forkert placeret)	

Procedure Hjaelp kan kaldes mens spillet er i gang. Det komplicerede i procedure er selvfølgelig ikke at få skrevet tekstskærmen op, men at få reetableret skærmen, når spillet skal gå videre.

Slet skærmen og tegn en ramme
Skriv hjælpeteksten
Vent på ENTER
Slet skærmen og tegn en ramme
Tæl alle de foregående gæt igennem
Skriv gæt nr. for det nummer og tekst
Tegn gættet for det nummer
Skriv hvor mange rigtige der er i gættet for det nummer
Skriv gæt nr. for det gæt man er i gang med

Procedure Initialiser skal sætte betingelserne for spillet op. Som parameter får den sværhedsgraden, så betingelserne sættes op efter det. Herefter genereres det skjulte gæt.

Sæt antallet af brikker op efter den valgte sværhedsgrad
Sæt antallet af farver op efter den valgte sværhedsgrad
Sæt op om der må være ens efter den valgte sværhedsgrad
Sæt det aktuelle gæt til nr. 0
Start tilfældigtals generatoren
Tæl alle brikkerne igennem
Lav et gæt for den aktuelle brik
Er det tilladt med ens brikker? Ja / Nej
Sæt op at der ikke er andre af det aktuelle gæt
Tæl de foregående gæt igennem
Er det aktuelle gæt = et af de foregående? Nej / Ja
Sæt op at der er andre af det aktuelle gæt
Er der andre af det aktuelle gæt? Nej / Ja
Lav et gæt for den aktuelle brik
Indtil der ikke er andre af det aktuelle gæt
Sæt den gættede farve på brikken til blå

Interface til Master Mind delen af programmet.

Proceduren `mm` ligger i uniten `mm_unit`. Proceduren har ingen parametre.

Uniten selv anvender funktionen `svaerhed`, der ligger i uniten `svaer`. Funktionen kaldes med en parameter, der angiver max sværhedsgrad, og funktionen returnerer et tal fra 1 til max, for sværhedsgraden. Hvis brugeren er kommet skævt ud af `svaerhed` kommer der 0 retur.

Konstanter brugt i programmet.

Konstanter der definerer rammerne for programmet

`max_gaet = 12;`	Det maksimale antal gæt på skærmen, og dermed i spillet.
`tot_brik = 5;`	Det største tal antal brikker kan sættes til.
`tot_farv = 7;`	Det største tal antal farver kan sættes til.
`max_svaer = 8;`	Antallet af sværhedsgrader der er i spillet.

Variabler brugt i programmet.

Der skal bruges et antal sæt af variabler, der kan indeholde et sæt brikker.

`rigtig : array [1..tot_brik] of byte;`	Indeholder den rigtige kombination.
`akt_gaet : array [1..tot_brik] of byte;`	Indeholder det aktuelle gæt.
`gl : array [1..max_gaet,1..tot_brik] of byte;`	Indeholder de gamle gæt (skal bruges hvis man har haft en hjælpeskærm fremme, til at skrive de gamle gæt op med).

Der skal bruges et antal variabler, der definerer begrænsningerne i spillet (antal brikker og antal farver m.v.). Disse skal indstilles afhængigt af sværhedsgraden.

`max_brik : byte;`	Antallet af brikker i den aktuelle sværhedsgrad.
`max_farv : byte;`	Antallet af farver i den aktuelle sværhedsgrad.
`ens : boolean;`	Om det er tilladt med ens farver i gættet.

Der skal bruges nogen arbejdsvariable, mens man gætter.

`brik : byte;`	Angiver hvilken brik man arbejder med.
`gaet_nr : byte;`	Angiver hvilket gaet man er i gang med.
`faerdig : boolean;`	Markerer om spillet er færdigt.
`N : byte;`	Generel tællevariabel.

Interne procedurer og funktioner brugt i programmet.

`Procedure Ramme`	Sletter skærmen og laver en ramme omkring skærmen, hvor der står MASTER MIND i toppen.
`Procedure W_3y`	Skriver teksten på den ønskede linie i position 3.
`Procedure W_gaet`	Skriver det ønskede gæt på den ønskede linie på skærmen i position 15.
`Procedure Start_skaerm`	Stiller en velkomstskærm op og forbereder til indtastning af sværhedsgraden
`Function Test_rigtig`	Tester om gættet er rigtigt, beskrevet som NS-diagram. Returnerer sandt, hvis gættet er rigtigt.
`Procedure Hjaelp`	Opbygger hjælpeskærmen og venter på ENTER. Herefter reetableres skærmen med de gæt der er lavet.
`Procedure Initialiser`	Stiller betingelserne op for spillet ud fra den valgte sværhedsgrad. Laver det rigtige (skjulte) gæt.
`Procedure Sorry`	Skriver en skærm med en beklagende meddelelse.
`Procedure Tillykke`	Skriver en skærm med alle mulige lykønskninger, der kan klemmes ind på et skærmbillede.
`Procedure mm`	Hovedproceduren, der holder styr på hele programmet.
`Procedure Ss`	Stiller .

Konklusion.

Spillet Master mind er kommet til at fungere fint i et DOS-miljø.
Der er fundet et brugerinterface der virker fornuftigt, og det virker umiddelbart let at gå til for andre.
Spillet er testet som en unit, der kan afvikles fra et testprogram, men det er endnu ikke afprøvet sammen med andre spil.

Bilag.

Bilag 1. Selve koden.

```
unit mm_unit;

interface  { Det der skal kunne kaldes fra andre moduler. }

Procedure mm;

function e_read(var indt : integer) : boolean;


implementation   { Koder der laver noget i det her modul. }

uses crt, svaer;

const
  max_gaet = 12;
  tot_brik = 5;
  tot_farv = 7;
  max_svaer = 8;

  pil_op   = #72;
  pil_ned  = #80;
  pil_ve   = #75;
  pil_hoe  = #77;
  F9       = #67;
  F1       = #59;

  sv_arr : array [1..max_svaer,1..3] of byte =
 { Brikker, Farver, ens}
   ( (3,      4,     0),
     (3,      4,     1),
     (3,      6,     0),
     (3,      6,     1),
     (4,      5,     0),
     (4,      5,     1),
     (5,      7,     0),
     (5,      7,     1) );
var
  rigtig   : array [0..tot_brik] of Byte;
  akt_gaet : array [0..tot_brik] of Byte;
  gl       : array [1..max_gaet,0..tot_brik] of Byte;

  max_brik : byte;
  max_farv : byte;
  ens      : boolean;

  gaet_nr  : byte;
  brik     : byte;
  N, I     : byte;
  faerdig  : boolean;
```

```
{ Indtastningsrutine til heltal, der laver fejltjek på indtastningen. }
function e_read(var indt : integer) : boolean;

var
  ch : char;
  n  : byte;
  st : string;
  code : integer;

begin
  n := 0;
  st := '';
  e_read := true;
  repeat
    ch := readkey;
    write(ch);
    case ch of
      #27 : begin
              e_read := false;
              exit;
              end;
      #13 : ;
      #8  : if n = 0 then write(#7)
               else begin
                 write(' ',#8);
                 dec(n);
                 st := copy (st,1,length(st)-1);
                 end;
      '0'..'9','-' : begin
                       st := st + ch;
                       inc(n);
                       end;
      else write(#7,#8,' ',#8);
      end;
    until ch = #13;
  val(st, indt, code);
  end;

{ Renser skærmen og laver en nydelig ramme rundt om. }
Procedure Ramme;

var
  n : byte;

Begin
  textbackground(black);
  textcolor(yellow);
  clrscr;
  write('É'); for n := 1 to 77 do write('Í'); write('»');
  for n := 2 to 24 do begin
    gotoxy(1,n);
    write('º');
    gotoxy(79,n);
    write('º');
    end;
  gotoxy(1,25);
  write('È'); for n := 1 to 77 do write('Í'); write('¼');
  gotoxy(27,1);
  write('  M A S T E R    M I N D  ');
  end;
```

```
{ Skriver teksten fra 3. position på skærmen på linie y. }
Procedure W_3y ( y: byte; tekst : string );

Begin
  gotoxy(3,y);
  write(tekst);
  end;


{ Skriv gættet i parameteren på skærmen som farvede brikker. }
{ Parameteren y angiver hvor. }
Procedure W_gaet (y : byte; var gaet : array of byte);

var
  n: byte;

Begin
  for n := 0 to max_brik do begin
    gotoxy(n*2+15,y);
    textbackground(gaet[n]);
    write(' ');
    textbackground(0);
    write(' ');
    end;
  end;


{ Velkomstskærm, der forklarer hvem hvad hvordan, men absolut ikke hvorfor. }
{ Herefter vælges sværhedsgraden. }
Procedure Start_Skaerm;

var
  n : byte;

Begin;
  Ramme;
  textcolor(cyan);
  W_3y(3, 'Velkommen til Master Mind');
  W_3y(5, 'Du skal gætte en kombination af farvede brikker');
  W_3y(6, 'Ved hvert gæt får du at vide hvor mange brikker der er rigtig farve og');
  W_3y(7, 'rigtigt placeret, og du får at vide hvor mange derudover, der har den');
  W_3y(8, 'rigtige farve, men er forkert placeret.');
  W_3y(10,'Ønsker du hjælp til hvordan spillet spilles, trykkes F1 i spillet.');
  W_3y(12,'Spillet kan gennemføres med forskellige sværhedsgrader, som følger:');
  for n := 1 to max_svaer do begin
    W_3y(12+n, 'Sværhed');
    write(n:3);
    write(sv_arr[n,1]:4, ' brikker');
    write(sv_arr[n,2]:3, ' farver');
    if sv_arr[n,3] = 1
      then write(' med ens')
      else write(' uden ens');
    end;
  gotoxy(3,23);
  end;
```

```
{ Funktion der tester om gættet i parameteren er rigtigt. }
{ Parameteren y angiver hvor på skærmen der skal skrives. }
Function test_rigtig (y:byte; var gaet : array of byte) : boolean;

var
  n, i    : byte;
  korrekt : byte;
  farve   : byte;
  talt    : array [0..tot_brik] of boolean;
  g_talt  : array [0..tot_brik] of boolean;

Begin
  test_rigtig := true;
  korrekt := 0;
  farve := 0;
  { Her tjekkes om alle er rigtige, og der registreres hvilke der er korrekte }
  for n := 0 to max_brik do begin
    talt[n] := false;
    g_talt[n] := false;
    if gaet[n] <> rigtig[n]
      then test_rigtig := false
      else begin
        talt[n] := true;
        g_talt[n] := true;
        korrekt := korrekt + 1;
        end;
    end;
  { Her tjekkes hvilke der er rigtig farve, men forkert placeret }
  for n := 0 to max_brik do if not talt[n] then begin
    for i := 0 to max_brik do if not g_talt[i] then begin
      if rigtig[n] = gaet[i] then begin
        talt[n] := true;
        g_talt[i] := true;
        farve := farve + 1;
        i := max_brik;
        end;
      end;
    end;

  { Her udskrives hvor mange af hver slags der er }
  gotoxy(26,y);
  write(korrekt:5);
  if korrekt = 1 then write(' korrekt ')
    else write(' korrekte');
  write (farve:4);
  if farve = 1 then write(' farve ')
    else write(' farver');

  { Testudskrift
  for n := 0 to max_brik do if talt[n] then write(' 1') else write(' 0');
  write('  ');
  for n := 0 to max_brik do if g_talt[n] then write(' 1') else write(' 0');
  }
  end;
```

```
{ Procedure der skriver en hjælpeskærm, og når den er læst, reetableres }
{ skærmbilledet som det var ud fra de gamle gæt. }
Procedure Hjaelp;
var
  n     : byte;
  dummy : boolean;
Begin
  Ramme;
  W_3y( 2, '                                  H J Æ L P');  textcolor(3);
  W_3y( 4, '<-    vælger brikken til venstre for (hvis der er flere).');
  W_3y( 5, '->    vælger brikken til højre for (hvis der er flere).');
  W_3y( 6, chr(24)+'     vælger farven før den valgte.');
  W_3y( 7, chr(25)+'     vælger farven efter den valgte.');
  W_3y( 8, 'Esc   afslutter spillet i utide.');
  W_3y( 9, 'F1    giver denne hjælpeskærm.');
  W_3y(10, 'Enter godkender farvekombinationen og går videre i spillet.');
  W_3y(12, 'Tast Enter for at komme tilbage til spillet.');
  readln;

  { Skærmen slettes og alt hvad der stod der før hjælpeskærmen skal skrives igen }
  Ramme;
  textcolor(15);
  { Vis de forgående gæt. }
  for n := 1 to gaet_nr - 1 do begin;
    gotoxy(3,n*2);
    Write('Gæt nr.', n:3, '  ');
    W_gaet(n*2, gl[n]);
    dummy := test_rigtig(n*2, gl[n]);
    end;
  { Vis det aktuelle gæt. }
  gotoxy(3,gaet_nr*2);
  Write('Gæt nr.', gaet_nr:3, '  ');
  W_gaet(gaet_nr*2, akt_gaet);

  end;

{ Procedure der sætter alle de grundlæggende variabler i spillet op, laver }
{ det skjulte gæt og nulstiller det indtastede gæt. }
Procedure Initialiser(Svaer : byte);
var
  brugt : boolean;
  n, i  : integer;
Begin
  if (svaer < 0) or (svaer > max_svaer) then svaer := 1;

  { Her adskilles ud fra sværhedsgrad }
  max_brik := sv_arr[svaer, 1] - 1;
  max_farv := sv_arr[svaer, 2];
  ens      := (sv_arr[svaer, 3] = 1);
  gaet_nr  := 0;

  randomize;
  for n := 0 to max_brik do begin;
    rigtig[n] := random(max_farv) + 1;
    { Her genereres igen på samme brik, hvis ens = false }
    if not ens then begin
      repeat
        brugt := false;
        for i := 0 to n-1 do begin
          if rigtig[n] = rigtig[i] then brugt := true;
          end;
        if brugt then rigtig[n] := random(max_farv) + 1;
        until not brugt;
      end;
    akt_gaet[n] := 1;
    end;
  end;
```

```
{ En procedure der er lavet udelukkende for at håne folk der ikke kan }
{ finde ud af at spille mastermind. }
Procedure Sorry;

Begin
  Ramme;
  textcolor(cyan);
  w_3y(3, 'Sorry din tumpe, du kunne ikke nå det inden for ');
  write(Max_Gaet, ' gæt,');
  W_3y(4, 'så nu slutter festen');
  W_3y(6, 'Tast ENTER');
  faerdig := true;
  end;


{ En procedure der markerer at man har vundet, og hvor mange gæt man brugte }
Procedure Tillykke;

Begin
  Ramme;
  textcolor(red);
  for n := 0 to 2 do begin
    gotoxy(3,n+3);
    for i := 0 to 7 do write('Tillykke ');
    end;
  W_3y(7, 'Du nåede det på ');
  write(Gaet_nr, ' gæt.');
  for n := 0 to 2 do begin
    gotoxy(3,n+9);
    for i := 0 to 7 do write('Tillykke ');
    end;
  W_3y(13, 'Tast ENTER');
  faerdig := true;
  end;
```

```
{ mm er hovedproceduren i dette modul, det er den der kører hele spillet   }
{ Master Mind. Først initialiseres, herefter stilles op til et gæt, så     }
{ håndteres indtastningen, og til slut tjekkes om gættet var OK, og om man }
{ er færdig med spillet. }
Procedure mm;
var
  ch      : char;
Begin
  Start_Skaerm;
  n := Svaerhed(max_svaer);
  if n > 0 then begin
    Initialiser(n);
    ramme;
    faerdig := False;
    repeat { Mens der endnu er gæt endnu }
      gaet_nr := gaet_nr + 1;
      gotoxy(3,gaet_nr*2);
      textcolor(15);
      Write('Gæt nr.', gaet_nr:3, '  ');
      brik := 0;
      repeat { mens gættet vælges }
        W_gaet(gaet_nr*2, akt_gaet);
        gotoxy(brik*2+15,gaet_nr*2);
        ch := readkey;
        case ch of
          { Esc : Afbryder spillet }
          #27 : begin
                gotoxy(3,23);
                Write('Du afslutter nu spillet uden at fuldføre det.   Tast ENTER');
                { Hvis man alligevel ikke vil afbryde, kan man fortryde med }
                { alt andet end Enter }
                ch := readkey;
                if ch = #13 then begin
                  faerdig := true;
                  clrscr;
                  end;
                gotoxy(3,23);
                Write('                                                                  ');
                end;
          { Enter : Accepterer gættet som det er nu }
          #13 : for n := 0 to max_brik do gl[gaet_nr,n] := akt_gaet[n];
          { Alle specialtaster }
          #0 : begin
               ch := readkey;
               case ch of
                 pil_op  : akt_gaet[brik] := (akt_gaet[brik] mod max_farv) + 1;
                 pil_ned : if akt_gaet[brik] = 1
                             then akt_gaet[brik] := max_farv
                             else akt_gaet[brik] := akt_gaet[brik] - 1;
                 pil_hoe : if brik < max_brik then brik := brik + 1;
                 pil_ve  : if brik > 0 then brik := brik - 1;
{ snyd           F9      : for n := 0 to max_brik do akt_gaet[n] := rigtig[n]; }
                 F1      : hjaelp;
                 else write(#7); { Beep ved fejlindtastning }
                 end;
               end;
          else write(#7); { Beep ved fejlindtastning }
          end;
        until ch = #13; { Til det aktuelle gæt er afsluttet }
        if not faerdig then begin
          if test_rigtig (gaet_nr*2, akt_gaet)
            then Tillykke
            else if gaet_nr = max_gaet then Sorry;
          end;
      until faerdig;
    end;
  end;
end.
```

Bilag 2. Test Program, brugt til at afprøve mm_unit med.

```
program test;

uses mm_unit;

Begin
  mm;
  readln;
  end.
```

Bilag 3. Test-unit der var nødvendig for at afprøve programmet.

```
unit svaer;

interface

Function svaerhed(Max_Svaer : byte) : byte;

implementation

uses crt, mm_unit;

Function Svaerhed(Max_Svaer : byte) : byte;

var
  temp : byte;

begin

  Write('Indtast Sværhedsgrad : ');
  e_read(temp);
  svaerhed := temp;

  end;

end.
```