Demo-Programme des Pascal/MATLAB-Kurses WS 2001, Gerhard Opfer

Programme mit Endung .p: Pascal-, mit Endung .m: MATLAB-Programme

Inhaltsverzeichnis

Hinweis: Die Zuordnung der Programme zu den Stunden ist nur in etwa richtig. Nicht alle Programme sind in der Vorlesung vorgekommen. Dieser Text und alle Programme stehen in

http://www.math.uni-hamburg.de/home/opfer/aufgaben01w.html

Programm a01_01.p Minimalprogramm

```
1  program NAME;
2  begin
3  end.
```

Programm a01_02.p Struktur eines Programms

```
1  program NAME;
2  {An diese Stelle gehoeren sog. Vereinbarungen}
3  begin
4  {An dieser Stelle steht das eigentliche Programm}
5  end.
6
7  {Was zwischen geschweiften Klammern steht, ist ein Kommentar,
8  der vom Programm nicht beachtet wird.} (*Kommentare koennen auch,
9  wie an diesem Beispiel zu sehen, gekennzeichnet
10 werden, insbesondere, wenn geschweifte Klammern {} vorkommen*)
11
12 {Der NAME besteht aus einer (endlichen) Folge von
13 (a) Buchstaben: ab...zAB...Z
14     (52 Stueck, keine Umlaute, kein Esszett),
15 (b) Ziffern: 01...9 (10 Stueck),
16 (c) dem Unterstreichungszeichen: _
17 mit folgender Zusatzregel: An der ersten Stelle steht ein Buchstabe}
```

Programm a01_03.p Hallo-Programm mit `writeln` etc.

```
1  program Hallo; {Der Name (hier Hallo) hat
2                  fuer das Programm keine Bedeutung}
3  {Pascal/MATLAB-Kurs WS 2001, Gerhard Opfer}
4  {Wir schreiben ein erstes Programm,
5   das etwas tut.}
6  begin
7    writeln('Hallo!'); {ln steht fuer line}
8    writeln('Wie geht es Ihnen?');
9    write('Hallo! ');
10   writeln('Wie geht es Ihnen?');
11   write('Je nach Wetterlage.');
12 end.
13 {Regel: Nach writeln wird in eine
14  neue Zeile geschrieben, nach write nicht.}
15
16 {Ergebnis (hineinkopiert):
17 Hallo!
18 Wie geht es Ihnen?
19 Hallo! Wie geht es Ihnen?
20 Je nach Wetterlage.}
```

Programm a01_04.p Volumen und Oberfläche eines Quaders

```
1  program mit_Variablen;
2  {Pascal/MATLAB-Kurs WS 2001, Gerhard Opfer}
3  {Wir schreiben ein erstes Programm,
4   in dem Variable vorkommen.}
5  {Wir berechnen das Volumen
6   und die Oberflaeche eines Quaders (Mauerstein),
7   der durch drei Groessen a,b,c definiert ist.}
8  var a,b,c,V,F:real;
9     {Wir teilen dem Programm die
10     Variablennamen (hier a, b, c, V, F) und die Bedeutung (hier real als
11     besondere Form einer Computerzahl) mit}
12 begin
13   a:=24; b:=11.5; c:=5;
14     {Die Zuweisung erfolgt durch := Der Abschluss ist
15      notwendig ein Semikolon; Die Anordnung ist
16      nicht wichtig, also auch untereinander moeglich.}
```

```
17 V:=a*b*c;           {Volumen}
18 F:=2*(a*b+a*c+b*c); {Flaeche}
19     {Die Multiplikationszeichen muessen mitgeschrieben werden}
20 writeln('Das Volumen des Quaders betraegt ',
21          V,'(Kubikzentimeter)');
22 writeln('Die Oberflaeche des Quaders betraegt ',
23          F,'(Quadratzentimeter)');
24 writeln; {leere Zeile}
25 writeln('Das Volumen des Quaders betraegt ',
26           V:6:2,' (Kubikzentimeter)');
27 {Die Angaben hinter V sind Formatangaben: 6 Stellen insgesamt,
28  2 Stellen davon nach dem Komma, Beispiel unten}
29 writeln('Die Oberflaeche des Quaders betraegt ',
30           F:6:2,' (Quadratzentimeter)');
31 end.
32
33 {Ergebnis (hineinkopiert):
34 Das Volumen des Quaders betraegt  1.38000000000000e+03(Kubikzentimeter)
35 Die Oberflaeche des Quaders betraegt  9.07000000000000e+02(Quadratzentimeter)
36
37 Das Volumen des Quaders betraegt 1380.00 (Kubikzentimeter)
38 Die Oberflaeche des Quaders betraegt 907.00 (Quadratzentimeter)}
```

Programm a01_05.p Einfache Summenbildung

```
 1 program mit_Variablen;
 2 {Pascal/MATLAB-Kurs WS 2001, Gerhard Opfer}
 3 {Wir schreiben ein  Programm,
 4  in dem Variable durch Einlesen bestimmt werden.}
 5 var x,y,s:real;
 6 begin
 7   writeln('Zwei Zahlen hintereinander eintippen, danach enter!');
 8   readln(x,y);{Das Programm erwartet
 9      eine Eingabe von zwei Zahlen x, y in einer Zeile
10      ueber die Tastatur}
11   s:=x+y;
12   writeln('Summe = ', s:6:2);
13   writeln('Eine Zahl eintippen, danach enter!');
14   read(x);    {nur eine Zahl ueber die Tastatur}
15   writeln('Noch eine Zahl eintippen, danach enter!');
16   read(y);
17   s:=x+y;
18   writeln('Summe = ', s:6:2);
19   {readln;} {Dieser Befehl kann in manchen Systemen
20             gut zum Stoppen des Programms benutzt werden.}
21   writeln('Schluss');
22 end.
```

Programm a01_06.p Test, ob Namen der Art a___b c__ erlaubt sind (ja)

```
1 {Pascal/MATLAB-Kurs WS 2001, Gerhard Opfer}
2 {Test ob _ auch mehrfach und hinten vorkommen koennen}
3 program a_;
4 var x_____y,aa__:real;
5 begin
6 x_____y:=2.5;aa__:=1;
7 writeln(aa__,x_____y);
8 end.   {Programm funktioniert}
```

Programm a02_01.m Beispiele zu einfachen MATLAB-Techniken, insbesondere help-, lookfor-Funktionen und eval- und feval-Befehle.

```
%Einige einfache Beispiele zu MATLAB-Sprachelementen.
%Kommentare werden zeilenweise durch % (Prozentzeichen) eingeleitet.
%Nach dem Aufruf von MATLAB (Klicken bei windoofs-Rechnern,
%Tippen des Wortes "matlab" in ein Kommandofenster bei Suns)
%erscheint in jedem Fall ein Fenster mit einem 'Prompt' >>
%Danach werden Befehle sofort ausgefuehrt:
format loose %das Gegenteil von compact
x=3*4+5
disp('Viel Luft im Format: Enter-Taste druecken'); pause
%Will mann nicht soviel Luft zwischen den Zeilen, schreibe man
format compact
x=pi*4
disp('Luft ist weg und langes Format kommt: Enter-Taste druecken'); pause
%Moechte man mehr Stellen sehen, schreibe man
format long
x=pi*4
%Moechte man wieder zur kurzen Form zurueckkehren, schreibe man
format short
disp('diverse Rechnungen: Enter-Taste druecken'); pause
%Laengere Programme schreibe man in eine Datei der Form "datei.m"
%Dabei ist die Endung .m notwendig. Diese Programme werden
%als MATLAB-Programme identifiziert und mit Tippen von
%"datei" (ohne " ") und ohne Endung aufgerufen.
%Wesentlich in MATLAB sind Variablen vom Typ Matrix oder Vektor,
%wobei explizite Deklarationen nicht moeglich  sind.
%Vektoren sehen entweder so aus
x=[1 2 3 4 5] %Zeilenvektoren oder so
y=[1;2;3;4;5] %oder mit der gleichen Bedeutung
y=[1
   2
   3
   4
   5]; %Spaltenvektor
disp('Matrix: 2 Zeilen, 3 Spalten');
A=[1 2 3; 4 5 6]
%oder
A=[1 2 3
   4 5 6];
       %x kann aber im vorliegenden Fall auch einfach in der Form
x=1:5; %geschrieben werden. Man kann leicht einen
       %Zeilen- in einen Spaltenvektor umwandeln
y=x';  %Jetzt ist y ein Spaltenvektor wie oben.
       %Das Semikolon am Ende hat eine total neue Bedeutung:
       %Es verhindert das Ausgeben auf den Bildschirm.
y;     %allein bewirkt, dass y auf dem Schirm angezeigt wird.
       %Man kann Operationen auf alle Vektorkomponeneten gleichzeitig anwenden:
disp('x/3:');
x/3
disp('x+3:');
x+3
disp('x.^2:');
x.^2
disp('x.^x: Enter-Taste druecken'); pause
     %Operationen wie .* oder ./ oder .^ wirken komponentenweise
     %auf den ganzen Vektor und entsprechend auch
x.^x
disp('x./x: Enter-Taste druecken'); pause
x./x
disp('A.^2: Enter-Taste druecken'); pause
A.^2
     %Ist
x=0:0.01:2*pi;   %also x=0,0.01,0.02,...,2*pi, so ist
y=sin(x); %entsprechend der Vektor sin(0),sin(0.01),sin(0.02),...,sin(2*pi)
```

```
%Ein Zeichenbefehl hat dann die Foerm
disp('Sinus-Kurve in blau: Enter-Taste druecken'); pause
plot(x,y);
disp('Koennen Sie auch noch beim naechsten Mal die x-Achse einzeichnen?');
disp('2 Punkte verbinden: Enter-Taste druecken'); pause
%Will man zwei Punkte (x,y), (u,v) geradlinig verbinden, so ist
x=0;y=0;u=2;v=-1; %und
plot([x u],[y v]) %der geignete Befehl. Moechte man eine rote Linie,
disp('dieselben Punkte in rot verbinden: Enter-Taste druecken'); pause
%so schreibe man
plot([x u],[y v],'r');
disp('dasselbe gepunktet (suchen, wo sind die Punkte?): Enter-Taste druecken'); pause
plot([x u],[y v],'r.');
disp('Jetzt kommt etwas Langes, naemlich help plot: Enter-Taste druecken'); pause
%Ein wesentliche Informationsquelle sind die beiden Befehle
disp('help plot [langer Ausdruck]'); pause(3)
help plot %(plot nur als Beispiel) und
disp('Jetzt kommt lookfor cyan (dauert), Enter-Taste druecken'); pause
lookfor cyan
disp('cyan kommt also im Programm "cool" vor');
%help gibt Informationen ueber bestehende Programme,
%lookfor "text" sucht in den Kommentaren aller vorhandenen Programme
%(auch in den selbst geschriebenen) nach dem Wort "text"
%(ohne Anfuehrungszeichen).

%Ein wesentlicher Unterschied zu Pascal: Alle Schleifen und alle if-Abfragen
%muessen mit "end" (oder "end;") beendet werden. Als Schleifen gibt es "for"-
%und "while"-Schleifen, kein "repeat". Ein "case ... of" gibt es in der Form
%von "switch".

%Prozeduren muessen in separate Dateien name.m ausgelagert werden.
%Das .m am Ende ist obligatorisch. Aufgerufen werden sie nur mit dem
%Dateinamen name. Der Dateiname "name" sollte mit mit dem Prozedurnamen
%uebereinstimmen. Bei Abweichung wird der Dateiname genommen.
%Prozeduren haben immer die Form

%function [Ausgabeliste]=name(Eingabeliste).
%Anweisungsteil

%Die Eingabeliste wird nicht ueberschrieben, ist also nach
%Beendigung der Prozedur wieder mit den alten Werten besetzt.
%Der Ausgabeteil und auch der Eingabeteil koennen leer sein.

%Im Gegensatz zu Pascal, kann bei der Formulierung der Prozedur die Anzahl
%der eingehenden oder ausgehenden Parameter abgefragt werden mit:
%"nargin", "nargout". Strings koennen ueber eval(String) direkt ausgefuehrt
%werden, bei Funktionen mit feval(String,x),  beispielsweise
disp('Sinus an der Stelle 0.4 ueber einen String: Enter-Taste druecken'); pause
s='sin'; y=feval(s,0.4) %gibt y=sin(0.4);  %oder
for j=1:4
  s=['print -dpsc Bild',int2str(j),'.ps'];
  eval(s); %Es werden Dateien mit den Namen Bild1.ps,...,Bild4.ps erzeugt
end;
%Der Befehl int2str(j) wandelt die ganze Zahl j in einen String um,
%entsprechend arbeitet num2str(x) fuer eine reelle Zahl x.

disp('Probieren Sie zuerst im Dialogfenster alles Moegliche aus!');
disp('Schreiben Sie aber schliesslich Ihre Programme in Dateien.');
disp('Schluss der Demonstration!');
```

Programm a03_01.m Dyadische Summe als Übung zur Matrixbehandlung, vgl. auch a10_01.m.

```
%Bei MATLAB ist es wichtig, nach Moeglichkeit keine Schleifen zu verwenden
%Bei MATLAB fangen alle Matrizen (auch Vektoren) mit dem Index 1 an
%Beispie: Matrizenbesetzung B(j,k)=j+k,j,k=1,2,...,n (etwas trickreich),
```

```
%sog dyadische Summe
n=1000;

%erste Rechnung
tic %Stoppuhr wird gestartet
J=(1:n)';
A=J*ones(1,n);
B=A+A';
toc %Stoppuhr wird gestoppt und Zeit angezeigt
%elapsed_time = 1.3319 sec auf Sun ULTRA1
disp('erste Rechnung zu Ende');

%zweite Rechnung: sehr trickreich (vgl. a10_01.m, "Tonys Trick")
tic
J=(1:n)';
A=J(:,ones(n,1));
B=A+A';
toc
%elapsed_time = 0.7327 sec auf Sun ULTRA1
disp('zweite Rechnung zu Ende');

%dritte Rechnung, jetzt ohne Trick, wie in Pascal
tic
for j=1:n
 for k=1:n
  B(j,k)=j+k;
 end; %for k
end; %for j
toc
%elapsed_time =  42.5121 sec auf Sun ULTRA1
disp('dritte (laengste) Rechnung zu Ende');
```

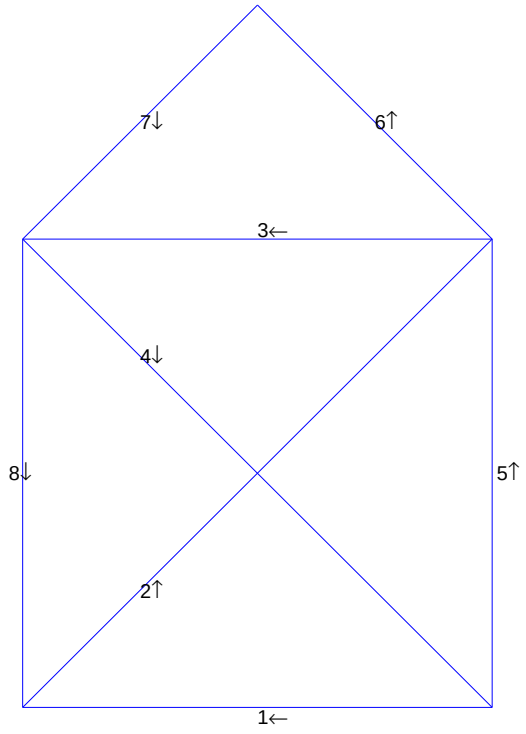
Programm a03_02.m Das Haus des Nikolaus, Übung für plot, mit Bild (Seite 6)

```
%Ein kleiner plot der das Haus des Nikolaus zeichnet.
%Pascal/MATLAB-Kurs Okt. 2000 und Okt. 2001, Gerhard Opfer
plot([0 1],[0 0]);hold on,    text(0.5,-0.02,'1\leftarrow');
axis([-0.2,1.2,0,1.5]),axis off, axis equal, pause(2)
plot([0 1],[0 1]);             text(0.25,0.25,'2\uparrow');pause(2)
plot([1 0],[1 1]);             text(0.5,1.02,'3\leftarrow');pause(2)
plot([0 1],[1 0]);             text(0.25,0.75,'4\downarrow');pause(2)
plot([1 1],[0 1]);             text(1.01,0.5,'5\uparrow');pause(2)
plot([1 0.5],[1 1.5]);         text(0.75,1.25,'6\uparrow');pause(2)
plot([0.5 0],[1.5 1]);         text(0.25,1.25,'7\downarrow');pause(2)
plot([0 0],[1 0]);             text(-0.03,0.5,'8\downarrow');pause(2)
                               text(0.3,-0.1,'Das Haus vom Nikolaus');
hold off;
print -dpsc nikolaus.ps  %erzeugt farbiges Bild als Datei nikolaus.ps
```

Programm a04_01.p case ... of und Funktionsprozedur

```
(*Wir demonstrieren die case ... of Anweisung an einem Beispiel*)
{Gleichzeitig wird noch einmal die Form einer Funktionsprozedur geuebt.
 Programm von Elsbeth Bredendiek}
{Pascal/MATLAB-Kurs Okt. 2000, Gerhard Opfer}
{Bei dem Programm geht es um die Frage, wieviele Tage die einzelnen
 Monate haben, wobei natuerlich nur der Februar interessant ist.}
program case_of;

function anz_tage(monat,jahr:integer):integer;
begin
  if (monat >0) and (monat <=12) then
  begin
    case monat of
      1,3,5,7,8,10,12:anz_tage:=31;
      4,6,9,11       :anz_tage:=30;
```

```
16       2              : if ((jahr mod 4=0)
17                         and (jahr mod 100 <> 0))
18                         or (jahr mod 400=0)
19                         then anz_tage:=29
20                         else anz_tage:=28;
21     end; {case ... of}
22   end else
23   begin
24     writeln('Monatseingabe ',monat:4,' falsch!');
25     anz_tage:=0;
26   end; {else}
27 end; {function}
28 {Prozeduren brauchen eine begin ... end;-Struktur}
29 begin {Hauptprogramm}
30   writeln(
31   anz_tage(15,2001):4,
32   anz_tage(2,1600):4,
33   anz_tage(2,1700):4,
34   anz_tage(2,1800):4,
35   anz_tage(2,1900):4,
36   anz_tage(2,1999):4,
37   anz_tage(2,2000):4);
38 end.
39 {Ergebnis:
40 Monatseingabe   15 falsch!
41    0  29  28  28  28  28  29}
```



Das Haus vom Nikolaus

Figur 4.1 Das Haus vom Nikolaus

Programm a04_02.p Sog. Collatz- oder $(3n+1)$-Funktion, Beispiele für Prozeduren auch vom Typ function

```
{Pascal/MATLAB-Kurs Okt. 2000, Gerhard Opfer}
{Wir beschaeftigen uns mit der sog. Collatz-Funktion. Wir schreiben ein Programm, das
bei gegebener  ganzer Zahl "Eingabe" > 1 ausrechnet, wie oft die Collatz-Funktion
iteriert werden muss, damit 1 heraus kommt. Gleichzeitig berechnen wir die groesste
dabei auftretende Zahl. Es gibt also einen Eingabewert und zwei Ausgabewerte.}

program collatz;
type neu=integer32; {longint in windows}
var rein,raus, maxi,zeler:neu;
{=============================================================================}
procedure Collatzfunktion (Eingabe: neu; var AusgabeSchrittzahl,AusgabeMaximum: neu);
{Wird eine Variable nicht mit "var" in der Liste deklariert (wie Eingabe),
 so hat sie nach Verlassen der Prozedur wieder den alten Wert.
 Es findet also eine Zwischenspeicherung statt.}

var Schritt, maxi, j: neu;
{============================}
{Im Vereinbarungsteil der Prozedur Collatzfunktion
 wird eine weitere Prozedur vereinbart}
function Collatz (i: neu): neu;
{ein Schritt der Collatz-Funktion}
begin
  if i = 1 then
  Collatz := 1
  else   {vor else kein Semikolon!}
  begin
    if i mod 2 = 0 then {durch 2 teilbar}
    Collatz := i div 2
    else          {nicht durch 2 teilbar}
    Collatz := 3 * i + 1;
  end;
end; {function Collatz}
{============================}
begin
  Schritt := 0;
  j := Eingabe;
  maxi := j;
  while j > 1 do {Dieser Test (statt ungleich 1) bewirkt,
                  dass  negative Eingabewerte nicht verarbeitet werden.}
  begin
    j := Collatz(j);
    Schritt := Schritt + 1;
    if j > maxi then
    maxi := j;
  end; {while}
  AusgabeSchrittzahl := Schritt;
  AusgabeMaximum := maxi;
end; {Collatzfunktion}
{=============================================================================}
begin {des eigentlichen Programms}        {Ergebnis (hineinkopiert)}
  zeler:=0;                               {rein  raus   max}
  writeln;                                {---------------}
  writeln('rein':6,'raus':6,'max':8);     {20      7     20}
  writeln('  ------------------');         {21      7     64}
  for rein:=1 to 3000 do                  {22     15     52}
  begin                                   {23     15    160}
    Collatzfunktion(rein,raus,maxi);      {24     10     24}
    {Ergebnisse nur fuer grosse Zahlen}   {25     23     88}
    if (raus>150) and (maxi <> 9232) then {26     10     40}
    begin                                 {27    111   9232}
      zeler:=zeler+1;                     {28     18     52}
      writeln(rein:6,raus:6,maxi:8);      {29     18     88}
    end; {if}                             {30     18    160}
```

```
64   end; {for}
65   writeln('  ------------------');
66   writeln('Insgesamt ',zeler:6,' Faelle');
67 end. {Hauptprogramm}
```

Programm a04_03.p Matrizenmultiplikation mit entsprechenden Deklarationen mit verschiedenen Typen von Matrixdefinitionen und Beispielen für Funktionsprozeduren, Übung zur type-Deklaration

```
 1 {Pascal/MATLAB-Kurs Okt. 2000, Gerhard Opfer, Ergaenzungen Okt. 2001}
 2 (*Wir berechnen das Matrix-Vektorprodukt
 3  y_j=sum_{k=1}^n a_jk*x_k},j=1,2,...,m*)
 4 (* sind auch Kommentarklammern, besonders wenn {} vorkommen*)
 5 program matrix_vektorprodukt;
 6 const mmax=200;nmax=300;
 7 type vektorm=array[1..mmax] of real;
 8      vektorn=array[1..nmax] of real;
 9      matrix=array[1..mmax,1..nmax] of real;
10      matrix_alternativ=array[1..mmax] of vektorn;
11 var  mm,nn:integer;
12      B:matrix;
13      B_alternativ:matrix_alternativ;
14      u:vektorm;
15      v:vektorn;
16 {================================================================}
17 function matrix_mal_vektor(m,n:integer;A:matrix;x:vektorn):vektorm;
18 var j,k:integer;
19     s:real;
20 begin
21   for j:=1 to m do
22   begin
23     s:=0;
24     for k:=1 to n do
25     s:=s+A[j,k]*x[k];
26     matrix_mal_vektor[j]:=s;
27   end; {for j}
28 end; {function}
29 {================================================================}
30 function matrix_mal_vektor_alternativ
31                 (m,n:integer;A:matrix_alternativ;x:vektorn):vektorm;
32 var j,k:integer;
33     s:real;
34 begin
35   for j:=1 to m do
36   begin
37     s:=0;
38     for k:=1 to n do
39     s:=s+A[j][k]*x[k];
40     matrix_mal_vektor_alternativ[j]:=s;
41   end; {for j}
42 end; {function}
43 {================================================================}
44 begin {Hauptprogramm}
45   mm:=3;nn:=2;
46   B[1,1]:=1;B[1,2]:=2;
47   B[2,1]:=3;B[2,2]:=4;
48   B[3,1]:=5;B[3,2]:=6;
49   v[1]:=-2; v[2]:=3;
50   u:=matrix_mal_vektor(mm,nn,B,v);
51   writeln(u[1]:6:0,u[2]:6:0,u[3]:6:0,u[4]:6:0);
52   {dasselbe mit dem alternativen Programm:}
53   B_alternativ[1][1]:=1;B_alternativ[1][2]:=2;
54   B_alternativ[2][1]:=3;B_alternativ[2][2]:=4;
55   B_alternativ[3][1]:=5;B_alternativ[3][2]:=6;
56   u:=matrix_mal_vektor_alternativ(mm,nn,B_alternativ,v);
57   writeln(u[1]:6:0,u[2]:6:0,u[3]:6:0,u[4]:6:0);
```

```
58 end.  {Hauptprogramm}
59 {================================================================}
60 {Ergebnis in beiden Faellen:
61  4     6     8     0   (0 ist zufaellig!)}
```

Programm a04_04.p Programm zum Testen von DIN 66256, Stufe 1. Wenn implementiert, können z. B. array-Grenzen abgefragt werden. Geht nicht auf Turbo- unf Free-Pascal, jedoch auf Sun- und Gnu-Pascal

```
 1 program Pascal_Norm_Stufe_1;
 2
 3 {Pascal/MATLAB-Kurs Okt. 2000, Gerhard Opfer}
 4 {DIN 66256 Stufe 1, Herschel & Pieper, 5. Aufl. S. 134/135}
 5 {Pruefen ob diese Stufe implementiert ist. Bei turbo nein, man erhaelt in der}
 6 {2. Programmzeile "procedure p(..."  eine Fehlermeldung,}
 7 {gibt auch Fehlermeldung bei "Free Pascal (Linux-Variante)}
 8 {bei Sun-Pascal: ja, Ergebnis s. u.funktioniert. Das Funktionieren}
 9 {dieser Stufe erlaubt das Abfragen von array-Grenzen, also sehr nuetzlich.}
10
11 var a:array[1..10] of real;
12     b:array[-5..5] of real;
13     m,n:integer;
14 procedure p(var x:array[unten..oben:integer] of real;var u,o:integer);
15 var j:integer;
16 begin
17   for j:=unten to oben do
18     x[j]:=j;
19   u:=unten; o:=oben {das letzte Semikolon kann entfallen}
20 end; {procedure}
21 begin {Hauptprogramm}
22   p(a,m,n);
23   writeln(m:3,n:3,a[m]:3:0,a[n]:3:0);
24   p(b,m,n);
25   writeln(m:3,n:3,b[m]:3:0,b[n]:3:0);
26 end.  {Hauptprogramm}
27 {Ergebnis auf Sun-Pascal:
28   1 10  1 10
29  -5  5 -5  5
30 Funktioniert also auf Sun-Pascal}
```

Programm a05_01.p Programmwiederholungen mit `for`

```
 1 {Pascal-MATLAB-Kurs WS 2001, Gerhard Opfer}
 2 program Summe_1_bis_10;
 3 {Wiederholungen mit for}
 4 var Summe, Zahl: integer;
 5 begin
 6   Summe:=0;
 7   for Zahl:=1 to 10 do
 8     Summe:=Summe+Zahl;
 9   writeln('Summe = ',Summe:3); {muss 55 rauskommen}
10 end.
```

Programm a05_02.p Programmwiederholungen mit `while`

```
 1 {Pascal-MATLAB-Kurs WS 2001, Gerhard Opfer}
 2 program Summe_1_bis_10;
 3 {Wiederholungen mit while}
 4 var Summe, Zahl: integer;
 5     ch:char;
 6 begin
 7   Summe:=0;
 8   Zahl:=1;
 9   while Zahl <=10 do
10   begin
11     Summe:=Summe+Zahl;
```

```
12     Zahl:=Zahl+1;
13   end;
14   writeln('Summe = ',Summe:3); {muss 55 rauskommen}
15   {etwas interessanteres Beispiel:}
16   writeln('Mehrere Zeichen hintereinander eintippen, auch Leerzeichen');
17   read(ch);
18   while ch=' ' do
19     read(ch);
20   writeln(ch);
21   {Es wird solange gelesen, bis ch kein Leerzeichen ist}
22   {Das erste Nichtleerzeichen wird ausgegeben}
23 end.
```

Programm a05_03.p Programmwiederholungen mit repeat

```
 1 {Pascal-MATLAB-Kurs WS 2001, Gerhard Opfer}
 2 program Summe_1_bis_10;
 3 {Wiederholungen mit repeat}
 4 var Summe, Zahl: integer;
 5 begin
 6   Summe:=0;
 7   Zahl:=1;
 8   repeat
 9     Summe:=Summe+Zahl;
10     Zahl:=Zahl+1;
11   until Zahl > 10; {letzte addierte Zahl ist 10}
12   writeln('Summe = ',Summe:3); {muss 55 rauskommen}
13 end.
```

Programm a05_04.p Logische Elemente am Beispiel eines Schaltkreises

```
 1 {Pascal-MATLAB-Kurs WS 2001, Gerhard Opfer, aus dem Arsenal von
 2 Elsbeth Bredendiek}
 3 program Lampe;
 4 {In einem Stromkreis ist ein Schalter A  hintereinder geschaltet}
 5 {mit zwei parallelen Schaltern B und C. Befindet sich in dem}
 6 {                            ____B\____                  }
 7 {                           |          |                 }
 8 {     ______________A\______|          |______x_____     }
 9 {    |                      |          |            |    }
10 {    |                      |____C\____|            |    }
11 {    |                                              |    }
12 {    |___________ ~ _______________________________|    }
13 {Stromkreis eine Lampe ~, so soll in Abhaengigkeit von den Schalterstellungen}
14 {der drei Schalter ausgerechnet werden, ob die Lampe leuchtet oder nicht.}
15 {Bei x ist eine Stromquelle.}
16
17 var leuchtet, A_zu, B_zu, C_zu: boolean;
18     zA, zB, zC:char;
19 begin
20   writeln('Stromkreis: 3 Schalterstellungen eintippen.');
21   writeln('T oder t bedeutet: Schalter geschlossen, sonst offen.');
22   readln(zA, zB, zC);
23   {Schalterstellungen einlesen mit T oder mit t oder anderem Zeichen}
24   if (zA='T') or (zA='t') then A_zu:=true else A_zu:=false;
25   if (zB='T') or (zB='t') then B_zu:=true else B_zu:=false;
26   if (zC='T') or (zC='t') then C_zu:=true else C_zu:=false;
27   leuchtet:=A_zu and (B_zu or C_zu);
28   writeln('A geschlossen':15,
29           'B geschlossen':15,
30           'C geschlossen':15,
31           'Lampe leuchtet':15);
32   writeln(A_zu:15,B_zu:15,C_zu:15,leuchtet:15);
33 end.
34 {Ein Ergebnis:
35 STT
36   A geschlossen  B geschlossen  C geschlossen Lampe leuchtet
```

```
          false           true           true          false
Wie man sieht, lassen sich boolesche Variable auch ausgeben}
```

Programm a05_05.p Ein- und Ausgabe in und aus Dateien, gebraucht wird Datei a05_05.erg zum Lesen

```
{Pascal/MATLAB-Kurs Okt. 2000 und Okt. 2001, Gerhard Opfer}
program schreiben_und_lesen_und_testen;
{Das Lesen aus Dateien und das Schreiben in Dateien wird geuebt.}
{Gebraucht wird eine datei a05_05.erg aus der gelesen wird.
Erzeugt werden Dateien Datei1.erg und Datei2.erg in die geschrieben wird.}
var dat1,dat2,dat3:text;
    x,y,z:real;
begin
  rewrite(dat1,'Datei1.erg');{Datei anlegen und oeffnen}
  {in windows etwas andere Technik:}
  {assign(dat1,'Datei1.erg')}
  {rewrite(dat1);}
  rewrite(dat2,'Datei2.erg');{Datei anlegen und oeffnen}
  reset(dat3,'a05_05.erg');  {bestehende Datei oeffnen }
  read(dat3,x);read(dat3,y);read(dat3,z);readln(dat3);
  write(dat2,x,' ');write(dat2,y,' ');writeln(dat2,z,' ');
  {In dat2 wird nebeneineinder geschrieben (mit Luecke)}
  close(dat2);
  readln(dat3,x,y,z);
  writeln(dat1,x);writeln(dat1,y);writeln(dat1,z);
  {In dat1 wird untereineinder geschrieben}
  close(dat1);close(dat3);
end.
```

Programm a06_01.p type-Deklarationen, insbesondere von Ordinaltypen (Aufzähltypen), auch in arrays

```
program type_deklarationen;
{Variable werden in der Form var ... deklariert.
In der auf var folgenden Liste koennen vorgefertigte, standardisierte
oder selbst definierte Typen vorkommen, die ueber type ...
definiert werden. Zu den standardisierten Typen gehoeren (fast immer):
char
boolean
real
integer
text
string (nicht Standard)
array ... of
set of
case ... of
record ... end;
}
{Wir beginnen mit einigen type-Deklarationen, vor var ... anzuordnen}
type Farben=(rot,gruen,blau,schwarz,weiss,magenta,cyan);
     Attribute=(schoen,haesslich,gut,schlecht,fein,grob,modern,altmodisch,
                farbig,farblos,bunt);
     Geschlecht=(maennlich,weiblich,neutral);
     Urteil=array[rot..cyan] of Attribute;
var  f:Farben; a:Attribute; g:Geschlecht; u:Urteil;

function Eigenschaften(F:Farben):Attribute;
var x:Attribute;
begin
  case F of
    rot: x:=bunt;
    gruen: x:=farbig;
    blau: x:=altmodisch;
    schwarz,weiss: x:=farblos;
    magenta,cyan: x:=modern;
```

```
  end; {case of}
  Eigenschaften:=x;
end; {Eigenschaften}

begin {Hauptprogramm}
f:=gruen; a:=schoen;
u[pred(f)]:=a;
u[succ(magenta)]:=modern;
a:=u[succ(magenta)];
{Bei Ordinaltypen sind immer succ (Nachfolger)
 und pred (Vorgaenger) definiert.}
g:=neutral;
writeln(u[rot],' ',a);
writeln(Eigenschaften(schwarz),' ',g);

end.  {Hauptprogramm}
{Ergebnis
schoen modern
farblos neutral
}
```

Programm a06_02.p Berechnung von π mit Zufallszahlen, Übung zur Prozedurbildung und zu bedingten Anweisungen (if)

```
program pi;
{Die Zahl pi soll experimentell bestimmt werden.}
{Man erzeugt n gleichverteilte Paare (x,y) von Zufallszahlen im Quadrat
Q:=[0,1]^2 und prueft, wieviele, sagen wir m, davom im Viertelkreis liegen.
Das Quadrat hat die Flaeche Eins und der Viertelkreis die Flaeche pi/4.
Daher ist m/n approx =pi/4, also approximativ pi=4m/n}
{Uebung zur Prozedurbildung und zu bedingten Anweisungen (if)}

function piapprox(Anzahl_der_Versuche:integer):real;

  function drin(x,y:real):boolean;
    {Es wird festgestellt, ob (x,y) im Einheitskreis liegt}
    begin
      if sqr(x)+sqr(y)<1 then drin:= true else drin:=false;
    end; {function drin}

  var j,Treffer:integer;
      x,y,xx:real;
  begin
    Treffer:=0;
    for j:=1 to Anzahl_der_Versuche do
    begin
      x:=random(xx);y:=random(xx); {xx ist Sun-Macke, wird nicht gebraucht}
      if drin(x,y) then Treffer:=Treffer+1;
    end; {for}
    piapprox:=4*Treffer/Anzahl_der_Versuche;
  end; {function piapprox}

var piap, piexakt:real;
    j,wieoft:integer;
begin {Hauptprogramm}
  piexakt:=4*arctan(1);
  wieoft:=1;
  for j:=1 to 7 do
  begin
    wieoft:=wieoft*10;
    piap:=piapprox(wieoft);
    writeln(wieoft:8,piap:8:4,abs(piap-piexakt)/piexakt:10:6);
  end; {for}
end.  {Hauptprogramm}
{Ergebnisse: (die Genauigkeit ist nicht besonders gross)
  wieoft piaproxx rel Fehler
```

```
      10  2.8000  0.108732
     100  3.4400  0.094986
    1000  3.1040  0.011966
   10000  3.1500  0.002676
  100000  3.1374  0.001347
 1000000  3.1430  0.000437
10000000  3.1418  0.000052}
```

Programm a06_03.p Berechnung der Fakultät, auch rekursiv, Demonstration von integer-Überlauf, der rekursive Aufruf verliert immer

```
program Fakultaeten;
{Pascal/MATLAB-Kurs Oktober 2000, Gerhard Opfer}
{Berechnung von n! auf ueblichem und rekursivem Wege}
{Vorsicht: hat integer nur 16 Binaerstellen, geht das ab 8! schief}
(*type integer=integer16;{Simulation von 16 Binaerstellen}*)
var n,j,z:integer;

function faknormal(m:integer):integer;
var j,erg:integer;
begin
  erg:=1;
  for j:=2 to m do
    erg:=erg*j;
  faknormal:=erg;
end; {function}

function fakrekursiv(m:integer):integer;
begin  if m>1 then
  fakrekursiv:=fakrekursiv(m-1)*m   else
  fakrekursiv:=1;
end; {function}

function fakrekursivalt(m:real):real;
begin            {reelle Arithmetik!}
if m>1 then
  fakrekursivalt:=fakrekursivalt(m-1)*m
  else
  fakrekursivalt:=1;
end; {function}

begin {Hauptprogramm}
  for j:=1 to 13 do
    writeln(j:3,faknormal(j):12,fakrekursiv(j):12,fakrekursivalt(j):12:0);
  z:=clock; {Sun-Pascal: Zeit in Millisekunden}
  for j:=1 to 100000 do
    n:=faknormal(12);
  z:=clock-z; writeln('Zeit Normalberechnung ',z);
  z:=clock;
  for j:=1 to 100000 do
    n:=fakrekursiv(12);
  z:=clock-z; writeln('Zeit rekursive Berechnung ',z);
end. {Hauptprogramm}

{Ergebnisse: mit 32-stelliger Integer-Arithmetik (bis etwa 4295 Millionen)
  j          j!          j!          j!
         (integer     (integer     (real
           normal)    rekursiv)   rekursiv)
  1           1           1           1
  2           2           2           2
  3           6           6           6
  4          24          24          24
  5         120         120         120
  6         720         720         720
  7        5040        5040        5040
  8       40320       40320       40320
```

```
 9      362880      362880      362880
10     3628800     3628800     3628800
11    39916800    39916800    39916800
12   479001600   479001600   479001600
13  1932053504  1932053504  6227020800  (integer falsch: 13!=6 227 020 800)
Zeit Normalberechnung        717
Zeit rekursive Berechnung   1633 [das sind Millisekunden, rekursiv: schlecht]
mit 16-stelliger Integer-Arithmetik:
 1           1           1
 2           2           2
 3           6           6
 4          24          24
 5         120         120
 6         720         720
 7        5040        5040
 8      -25216      -25216    (8!=40320, oberhalb 32767=2^15-1)
 9      -30336      -30336}
```

Programm a06_04.p Übungen zu type-Deklaration mit set (Mengen), Mengenoperationen, auch Prozedurbildung

```
{Pascal/MATLAB-Kurs Oktober 2000 und 2001, Gerhard Opfer}
program mengen;
(*In Pascal sind auch Mengen erlaubt, die klassisch durch Aufzaehlung
in der Form Z:={1,2, 5, 10} oder F={rot,gruen,blau} geschrieben werden*)
(*Mengenoperationen:
verbal                          TeX             Pascal
-------------------------------------------------------------
Mengenbildung durch Aufzaehlung: {   }            [   ]
Vereinigung                       \cup             +
Durchschnitt                      \cap             *
Differenz                      \backslash          -
Teilmenge A in B               \subseteq           A<=B
Teilmenge B in A               \supseteq           A>=B
Element von                       \in              in
leere Menge                    \emptyset           []
Gleichheit                        =                =
Ungleichheit                    \not=              <>
-------------------------------------------------------------*)
type kl_Bu='a'..'z';
     gr_Bu='A'..'Z';
     Ziffer=0..9;
     Bit=0..1;
     Wort=set of kl_Bu; {alle Teilmengen mit kl. Buchstaben}
     Zahl=set of Ziffer;
     {Potenzmenge=set of Zahl;} {Menge von Mengen}
     {line 24 -  Set type must be range or scalar, not set, geht also nicht}
var w1,w2,w3,w4,w5,vokale,konsonanten: Wort;
    z1,z2,z3,z4: Zahl;

{Wieviele Elemente enthaelt eine Menge?}
function kard(x:Wort):integer;
var w:kl_Bu; j:integer;
begin
  j:=0;
  for w:='a' to 'z' do
    if w in x then j:=j+1;
  kard:=j;
end; {geht auch direkt mit card}

{Wie macht man eine Menge sichtbar?}
procedure sichtbar_machen(x:Wort);
var j:kl_Bu;
begin
  j:='a';
  while x<>[] do
```

```
46   begin
47     if j in x then
48     begin
49       write(j);
50       x:=x-[j];
51     end; {if}
52     j:=succ(j);
53   end; {while}
54   writeln;
55 end; {sichtbar_machen}
56
57 begin
58   w1:=[]; {leere Menge}
59   w2:=['a','c']; {Aufzaehlung}
60   w3:=['c','a'];
61   w4:=['a','b', 'd', 'f'];
62   z1:=[1,3,5,7,9];
63   z2:=[0,2,4,6,8];
64   vokale:=['a', 'e', 'i', 'o', 'u'];
65   konsonanten:=['a'..'z']-vokale; {Mengendifferenz}
66   writeln(kard(konsonanten):3);
67   sichtbar_machen(w1);
68   sichtbar_machen(w2);
69   w2:=w3+w4; {+Vereinigung;}
70   sichtbar_machen(w2);
71   sichtbar_machen(w4);
72   w5:=w4*w2; {*=Durchschnitt}
73   if w1 <= w4 then
74   sichtbar_machen(w5);
75   write('Alle Konsonanten: ');
76   sichtbar_machen(konsonanten);
77   sichtbar_machen(w3); {Ausgabe in alphabetischer Reihenfolge!}
78   {eine weitere Anwendung von Mengen in "lotto.p", s. Programm a11_01.p
79 end.
80
81 {Ergebnis:
82  21
83
84 ac
85 abcdf
86 abdf
87 abdf
88 Alle Konsonanten: bcdfghjklmnpqrstvwxyz
89 ac}
```

Programm a06_05.p Übungen zur type-Deklarationen, insbesondere zu Ordinaltypen, boolean, char, set, enthält Code-Tabelle zu char

```
1  {Pascal/MATLAB-Kurs Oktober 2000 und 2001, Gerhard Opfer}
2  program boolesche_Ausdruecke; {Tests mit diversen Typen, auch set}
3
4  (*In Pascal sind auch Mengen erlaubt, die klassisch durch Aufzaehlung
5  in der Form Z:={1,2, 5, 10} oder F={rot,gruen,blau} geschrieben werden*)
6
7  type Farben=(rot,blau,gruen,schwarz,weiss);
8  var b1,b2,b3:boolean;
9      x,y:real;
10     i,j:integer;
11     c:char;
12     F:Farben;
13     s:set of Farben;
14 begin
15   x:=1.2;y:=2;
16   b1:=x<y;  {true}
17   b2:=(x>0) and (y<=0); {false}
18   b3:=(b1=false); {Wg b1=true ist b3=false}
```

```
  writeln(b1,b2,b3);
  c:='A';
  writeln(succ(c),pred(succ(c))); {geht bei Ordinaltypen}
  s:=[schwarz,weiss];
  j:=card(s); {wieviele Elemente sind da drin?}
  writeln('Die Menge s enthaelt',j:3,' Elemente, naemlich');
  for F:=rot to weiss do
    if F in s then
      writeln(F);
  F:=gruen; writeln('Nachfolger von ',F,' ist ',succ(F));
  {F:=weiss; writeln('Nachfolger von ',F,' ist ',succ(F));}
  {kein Nachfolger da, gibt Fehlermeldung (s. u.) und leider auch dump}
  writeln('Nummer von ',F,' ist ',ord(F):2); {Numerierung startet bei Null!}
  writeln('Nummer von schwarz ist ',ord(schwarz):2);
  writeln('Die Grossen Buchstaben A...Z haben die Nummern:');
  for c:='A' to 'Z' do
    write(ord(c):3,' '); {hintereinander weg}
  writeln; {danach neue Zeile}
  writeln('Leerzeichen hat die Nummer',ord(' '):3);
  for i:=32 to 255 do
  begin
    write(i:3,' ',chr(i),' ');
    if (i mod 10 = 0) and (i>0) then writeln;
  end;
  writeln;
  write('Wie geht''s?');writeln; {Apostroph in der Ausgabe}
  {funktioniert}
end.
(*Ergebnisse:
truefalsefalse
BA
Die Menge s enthaelt  2 Elemente, naemlich
schwarz
weiss
Nachfolger von gruen ist schwarz
Nachfolger von weiss ist
Fehlermeldung:
Enumerated type value of 5 is out of range on output
Nummer von gruen ist  2
Nummer von schwarz ist 3
Die Grossen Buchstaben AB...Z haben die Nummern: 65 66 ... 90 (etwas geaendert)
 32    33 !  34 "  35 #  36 $  37 %  38 &  39 '  40 (
 41 )  42 *  43 +  44 ,  45 -  46 .  47 /  48 0  49 1  50 2
 51 3  52 4  53 5  54 6  55 7  56 8  57 9  58 :  59 ;  60 <
 61 =  62 >  63 ?  64 @  65 A  66 B  67 C  68 D  69 E  70 F
 71 G  72 H  73 I  74 J  75 K  76 L  77 M  78 N  79 O  80 P
 81 Q  82 R  83 S  84 T  85 U  86 V  87 W  88 X  89 Y  90 Z
 91 [  92 \  93 ]  94 ^  95 _  96 `  97 a  98 b  99 c 100 d
101 e 102 f 103 g 104 h 105 i 106 j 107 k 108 l 109 m 110 n
111 o 112 p 113 q 114 r 115 s 116 t 117 u 118 v 119 w 120 x
121 y 122 z 123 { 124 | 125 } 126 ~ 127  128  129  130
Rest bis 255 weggelassen, da Wiedergabe unvollstaendig
Wie geht's?
*)
{Anmerkung: Aus der mit Pascal richtig wiedergegebenen Tabelle
sind in obiger Tabelle nur noch die Zeichen vor 127 richtig. Z. B. ist:
196 A-Umlaut, 214 O-Umlaut, 220 U-Umlaut, 223 esszett,
228 a-Umlaut, 246 o-Umlaut, 252 u-Umlaut, das Leerzeichen hat Nr. 32.
Die obige Tabelle stimmt nur partiell mit einer 1990 in Cork verabschiedeten
256-Zeichen-Tabelle ueberein. (s. N. Schwarz: Einfuehrung in TeX, 3. Aufl.
Addison-Wesley, Bonn, 1991, 336 S., Tabelle auf S. 331)}
```

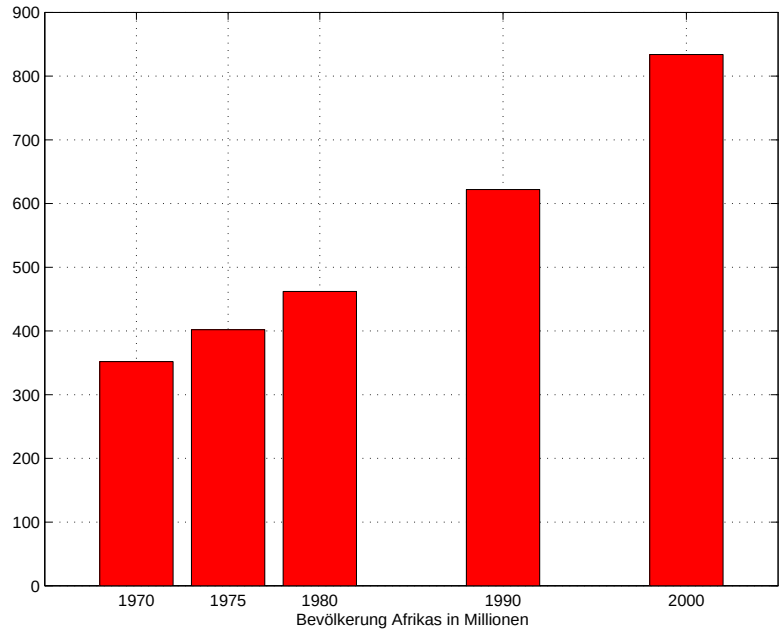
Programm a07_01.m Techniken zur Behandlung von plots, insbesondere Anbringung von Beschriftungen, Positionierung auch mit der Maus, auch Strichdicken-Manipulation und Erklärung sog. handle-Graphik, Verwendung von TEX-Zeichen

```
%Pascal/MATLABkurs Oktober 2000 und Oktober 2001, Gerhard Opfer

%Wir zeichnen einen Kreis und variieren das tatsaechliche Achsenverhaeltnis.
%Ausserdem bringen wir einige Beschriftungen an.
alpha=linspace(0,2*pi); %100 Punkte werden aequidistant verteilt
z=exp(i*alpha); %Gebrauch von komplexen Zahlen fuehrt zu einer
            %einfachen Anweisung, i=sqrt(-1) ist vordefiniert.
h=plot(z);  %wird unten gebraucht
%get(h);    %auch get(plot(z)) moeglich, umfangreiche Anzeige
disp('kein schoener Kreis, enter!'); pause
axis square %die Achsen werden unabhaengig von ihrer Beschriftung gleich lang
            %das Bild ist ein Quadrat
disp('axis square, enter!');
pause
axis equal  %das Bild ist rechteckig, aber die Massstaebe sind auf beiden
            %Achsen gleich
disp('axis equal, enter!');
pause
axis tight  %Der aeussere Rahmen ist jetzt eng anliegend
disp('axis zusaetzlich tight, enter!');
pause
axis([0,1,0,2]) %selbstgebastelte Achsenform, zuerst x- dann y-Achse
disp('axis([0,1,0,2]), enter!');
pause
disp('Aeusseren Rahmen dicker machen, enter!');
set(gca,'linewidth',4); pause
disp('Aeusseren Rahmen wieder in Normaldicke bringen, enter!');
set(gca,'linewidth','default'); pause
disp('Kurve dicker machen, enter!');
set(h,'linewidth',4); pause
%Das h kommt von oben, Zeile 8
disp('Kurve wieder in Normaldicke bringen, enter!');
set(h,'linewidth','default'); pause
close; %Fenster wird zugemacht
axis auto   %alte Form
clear all   %die Bedeutung und Besetzung aller Variablen werden eliminiert

%Beschriften kann man auch mit 'legend' machen. Um griechische Buchstaben
%und Hoeher- und Tieferstellungen zu bewirken, koennen Anleihen aus dem
%Satzsystem TeX uebernommen werden: \alpha, \beta, etc auch a^3 fuer
%Exponenten und x_2 fuer Indizes, \it fuer Kursivdruck.
%Dazu machen wir ein neues Bild:
disp('neues Bild');
t=0:63;
a=sin(2*pi/63*t); b=cos(2*pi/63*t);
stem(t,a+b); %so aehnlich wie plot aber mit 'Stamm'
hold on;
disp('stem-Graphik: sin+cos, enter!');
pause
plot(t,a,'r.');
disp('rot gepunkteter Sinus, enter!');
pause
plot(t,b,'g--');
disp('gruen gestrichelter Kosinus, enter!');
pause
legend('a+b','b=cos(x)','a=sin(x)');
disp('Legende angebracht, enter!');
pause
title('\it Legend stimmt nicht ganz');
xlabel('[0,2\pi]');
set(gca,'ytick',[-1.5:0.1:1.5]); %Neubeschriftung der y-Achse
disp('neue y-Achse, enter!');
```

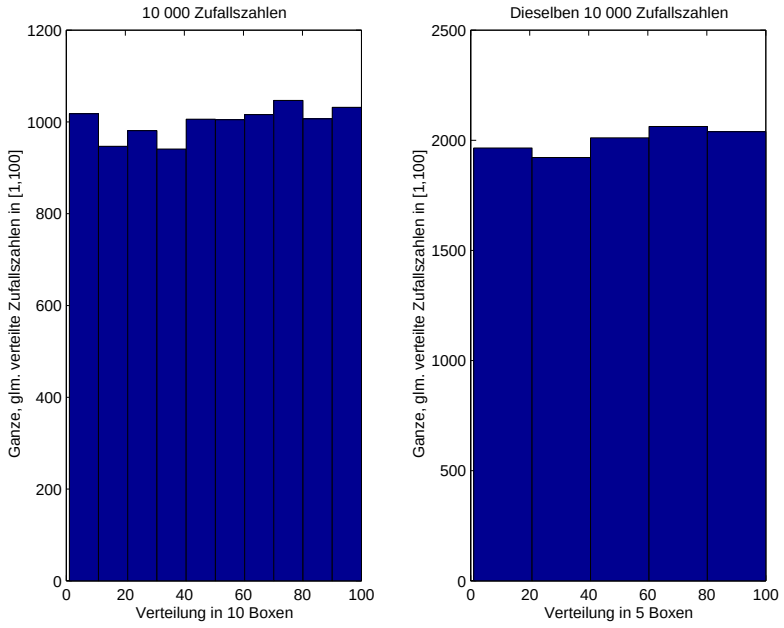
```
63 pause;
64 %Textposition (nicht der Text) kann auch mit Mausklick
65 %an die Zeichnung gebracht werden (sehr bequem):
66 disp('Text mit Maus positionieren (dreimal): sin + cos');
67 gtext('sin+cos');
68 disp('Text mit Maus positionieren: sin');
69 gtext('sin');
70 disp('Text mit Maus positionieren: cos');
71 gtext('cos');
72 hold off
```

Programm a07_02.m Vergleich von Histogrammen mit bar und mit hist, auch zwei Bilder

```
 1 %Wie zeichnet man ein Balkendiagramm (Histogramm)
 2 %Pascal/MATLAB-Kurs Okt. 2000, Gerhard Opfer
 3 %In MATLAB gibt es dazu (mindestens) zwei Befehle: 'hist' und 'bar',
 4 %die verschieden sch\"one Bilder produzieren und auch verschieden
 5 %zu bedienen sind. Wir w\"ahlen ein einfaches Beispiel:
 6 %Die Bev\"olkerung Afrikas in den Jahren
 7 %1970, 1975, 1980, 1990, 2000 betrug (bzw. wurde gesch\"atzt zu)
 8 % 352,  402,  462,  622,  834 Millionen Einwohner
 9 %(nach DTV Lexikon, 1999, Bd. 2, S. 244).
10
11 %Der Befehl 'bar' zeichnet genau das, was man naiverweise will. Man betrachte
12 %dazu das kleine Programm unten. Im vorliegenden
13 %Fall werden die Balken rot gezeichnet, wegen 'r' als drittem Parameter. Diesen
14 %Parameter kann man aber weglassen (dann werden die Balken blau gezeichnet).
15 %Weitere Einzelheiten unter 'help bar'.
16 %Wichtig ist nur, da\ss\ der Vektor Zeit strikt monoton geordnet ist
17 %(wachsend oder fallend). Der Befehl 'grid on' (geht auch bei 'plot')
18 %erzeugt ein Hintergrundgitter zur besseren Orientierung. Und 'xlabel(...)'
19 %funktioniert genau wie bei 'plot'.
20
21 %Der Befehl 'hist' rechnet ausgehend von einem (langen) Vektor y zun\"achst
22 %ymin=min(y) und ymax=max(y) aus und z\"ahlt dann, wieviele Elemente
23 %von y in die Intervalle
24 %[ymin+(j-1)*(ymax-ymin)/10,ymin+j*(ymax-ymin)/10[, j=1,2.,...,10
25 %fallen. Diese Anzahlen werden dann als Balken gezeichnet, und die x-Achse wird
26 %i. w. mit den Mittelwerten der Z\"ahlwerte in den einzelnen Boxen
27 %beschriftet. Die oben verwendete Zahl zehn kann hinter y durch
28 %eine andere Zahl ersetzt werden.
29 %F\"ur die Aufgabe 7 (Bev\"olkerungsentwicklung in den USA)
30 %ist also der Befehl 'bar' bestens geeignet.
31
32 %Demonstration von 'bar':
33 Zeit=[1970, 1975, 1980, 1990, 2000];
34 Bevoelkerung=[352, 402, 462, 622, 834];
35 figure(1);      bar(Zeit,Bevoelkerung,'r');    grid on;
36 xlabel('Bevoelkerung Afrikas in Millionen');    print -dpsc histo1.ps
37
38 %Demonstration von 'hist':
39 y=rand(1,10000);  %10000 auf ]0,1[ gleich verteilte Zufallszahlen.
40 y=floor(100*y+1); %10000 ganze, auf [1,100] gleichverteilte Zufallszahlen.
41 figure(2);    subplot(1,2,1);    hist(y);
42 %Die Syntax in den Klammern ist: m,n,j: das j-te Bild einer (m x n)-Matrix
43 title('10 000 Zufallszahlen');
44 ylabel('Ganze, glm. verteilte Zufallszahlen in [1,100]');
45 xlabel('Verteilung in 10 Boxen');
46 subplot(1,2,2);   hist(y,5);
47 %Farben k\"onnen hier nicht wie in 'bar' eingetragen werden
48 title('Dieselben 10 000 Zufallszahlen');
49 ylabel('Ganze, glm. verteilte Zufallszahlen in [1,100]');
50 xlabel('Verteilung in 5 Boxen');             print -dpsc histo2.ps
```



Figur 7.1. MATLAB-Histogramm mit `bar` (im Original rot)



Figur 7.2. MATLAB-Histogramme mit `hist` (im Original blau)

Programm a07_03.m Funktions-Prozedur zur rekursiven Berechnung der Fakultät

```
%program Fakultaet nur rekursiv
%Pascal/MATLAB-Kurs Oktober 2000 und 2001, Gerhard Opfer
%Berechnung von n! auf rekursivem Wege

function m=a07_03(n);
if n>1
  m=a07_03(n-1)*n;
  else
  m=1;
end;
```

Programm a07_04.m Aufruf von a07_03 und Vergleich mit direkter Berechnung

```
%Pascal/MATLAB-Kurs WS 2001, Gerhard Opfer
%Aufrufprogramm zur Fakultaet, normal und rekursiv
for n=[10,20,30,100]
  m=1:n;
  tic;         %Stoppuhr wird gestartet
  p1=prod(m);
  toc          %Stoppuhr wird gestoppt

  tic
  p2=a07_03(n);
  toc
end;

%       prod         rekursiv
%n= 10: 5.8100e-04   0.0021
%n= 20: 4.9300e-04   0.0044
%n= 30; 4.9500e-04   0.0053
%n=100: 5.0100e-04   0.1136
%Die rekursive Technik verliert immer
```

Programm a08_01.m Prinzip zur Herstellung von farbigen Rechtecksmustern, mit zufälligen Farbkombinationen und einem Schachbrett

```
%Pascal/MATLAB-Kurs Oktober 2001, Gerhard Opfer
%Es soll eine gegebene mit ganzen, positen Zahlen besetzte Matrix M
%als ein Farbarray gezeigt werden. Die Farben  selbst werden als
%Tripel [r g b] dargestellt mit 0<=r, g, b<=1. Die Zahlen darin bestimmem
%die Mischung aus den drei Farben rot (r), gruen (g), blau (b): also z. B.
%[1 0 0]: rot
%[0 1 0]: gruen
%[0 0 1]: blau
%[0 0 0]: schwarz
%[1 1 1]: weiss
%[1 1 0]: gelb
%[1 0 1]: magenta
%[0 1 1]: cyan
%[0.5 0.5 0.5]: grau
%alpha*[1 1 1]: Grautoene, 0<=alpha<=1
%alpha*[1 0 0]: Rottoene, 0<=alpha<=1, kleines alpha dunkelrot
%[1 0.62 0.62]: Kupfer
%[0.49 1 0.83]: Aquamarin

%Wir beginnen mit einem einfachen Beispiel:
M=[1 2 3; 4 5 6];
%Um den Zahlen 1 bis 6 eine Farbe zuzuordnen, definieren wir eine Farbtabelle
%bestehend aus 6 Zeilen und 3 Spalten entsprechend der obigen Farbdefinition.
%Also

Map=[1 0 0   %rot
     0 1 0   %gruen
     0 0 1   %blau
```

```
29      0 0 0   %schwarz
30      1 1 1   %weiss
31      1 1 0]; %gelb
32 zeit=3; %Laenge der Pause in Sekunden
33 %Der Vorgang ist immer derselbe, naemlich
34
35 %1. Herstellung einer (mxn)-Matrix M mit ganzzahligen Eintr\"agen
36 %   im Bereich [u,o] mit 1<=u<=o.
37 %2. Definition einer Farbtabelle Map in Form einer (o,3)-Matrix.
38 %   Die Zeile j in Map repr\"sentiert die Farbe j in der Matrix M.
39 %   Dann verwende man die entscheidende Befehlssequenz
40 %   image(M);colormap(Map);
41 %   Dann wird der Schirm (oder das Papier) unterteilt in mxn Rechtecke
42 %   und das kleine Rechteck an der Position (l,k) wird gef\"arbt
43 %   mit der durch j=M(l,j) definierten Farbe, die indirekt
44 %   durch die j-te Zeile von Map definiert ist.
45 image(M);colormap(Map); %Gibt hier ein 2x3 Farbmuster
46 disp('Vom Benutzer definierte Farben');pause(zeit)
47
48 %image(M) stellt die Matrix dar unter Verwendung der in Map
49 %angegebenen Informationen. Benutzt wird dazu die vorgefertigte
50 %Prozedur 'colormap', die den Zusammenhang herstellt.
51
52 %Es gibt auch vordefinierte Farbzuordnungen: Map=colorcube(n) definiert
53 %fuer die ganzen Zahlen von 1 bis n gewisse Farben.
54 image(M);colormap(colorcube(6)); %Ein Muster aus Grautoenen.
55 disp('Vorgefertigte Farbpalette "colorcube"');pause(zeit)
56
57 %Um ohne groesseren Aufwand grosse Farbmatrizen herzustellen, verwenden wir
58 %Zufallszahlen. Eine analoge Technik (mit anders definierten Matrizen)
59 %kann auch verwendet werden, um zweidimensionale  Einzugsbereiche des
60 %Newton-Verfahrens zu zeichnen, gibt schoene Bilder mit fraktalen Raendern.
61
62 mm=2; nn=3;
63 for j=1:3
64   figure(j)
65   m=10^(j-1)*mm; n=10^(j-1)*nn;  %zuerst (2,3), dann (20,30), dann (200,300)
66   M=round(63*rand(m,n))+1; %gibt ganze Zufallszahlen in [1,64]
67   Map=rand(64,3); %64 Zeilen mit jeweils drei Zufallszahlen in [0,1]
68   image(M); colormap(Map);
69   figure(j+3);
70   image(M); colormap(colorcube); %Dasselbe mit anderer Farbtabelle
71 end; %for
72 pause(5); close(1:6);
73
74 %Weiteres Beispiel: Schachbrett
75 disp('Schachbrett');
76 Schachbrett=zeros(8,8);
77 Schachbrett(1:2:7,1:2:7)=ones(4,4);
78 Schachbrett(2:2:8,2:2:8)=ones(4,4);%schachbrettartige Verteilung von 0 und 1
79 Schachbrett=Schachbrett+1;%schachbrettartige Verteilung von 1 und 2
80 map=[0,0,0;1,1,1]; %nur schwarz-weiss, Farbe1=schwarz.
81 image(Schachbrett);colormap(map); axis square;
82 xlabel('Schachbrett');
83 %Aufgabe: Man verwandele die senkrechten 1,...,8 in A,...,H
```

Programm a08_02.m surf als Darstellungsmittel für Matrizen

```
1 %Pascal/MATLAB-Kurs WS 2001, Gerhard Opfer
2 %Der fuer die Darstellung von Flaechen ueber dem R^2 konzipierte Befehl
3 %surf (von surface=Oberflaeche) kann gut zur Visualisierung wesentlicher
4 %Merkmale von Matrizen verwendet werden, insbesondere kann Bandstruktur
5 %gut erkannt werden.
6
7 n=12;
8 J=(eye(n)); %eye(n)=(nxn)-Einheitsmatrix
9 figure(1);
```

```
surf(J); view(-45,30); %Blickwinkel: Azimut und Hoehe
zlabel('Einheitsmatrix')
figure(2)
I=[zeros(n-1,1),eye(n-1);zeros(1,n)]; %Superdiagonale ist 1
K=[[zeros(1,n-1);eye(n-1)],zeros(n,1)]; %Subdiagonale ist 1
surf(0.8*I+J-2*K); view(-45,30);   %Blickwinkel
xlabel('3-Bandmatrix')
figure(3)
H=invhilb(round(n/2)); %inverse Hilbertmatrix, Eintraege ganzzahlig,
                       %Vorzeichen schachbrettartig,
surf(H); view(-37.5,0);
xlabel('Inverse Hilbertmatrix, frontal');
```

Programm a09_01.m Zeichnen von Kurven, am Beispiel von Zykloiden, Epizykloiden, Hypozykloiden; Zeichnen von Kurven im $\mathbb{R}^3$, auch sog. stem-plots, Einstellung des Blickwinkels, Hinweise auf demo-Programme

```
%Pascal/MATLAB-Kurs WS 2001, Gerhard Opfer
%Wir kennen gewoehnliche Funktionsplots (sinus, etc), wie zeichnet man aber
%parametrisierte Kurven? Beispiel: Zykloide. Welche Kurve macht eine
%bestimmte Stelle einer Fahrradspeiche beim Radfahren?
disp('Demonstration verschiedener Kurven und Darstellungstechniken');
Zeit=3;
figure(1)
t=linspace(-4*pi,4*pi);   rho=1;
for a=[0.25,1,4]
  x=rho*t-a*sin(t);
  y=rho-a*cos(t);
  plot(x,y,[t(1),t(length(t))],[0,0],'r');
  if a<rho
    xlabel('Gedehnte Zykloide');end;
  if a==rho
    xlabel('Gemeine Zykloide');end;
  if a>rho
    xlabel('Verschlungene Zykloide');end;
  axis equal,pause(Zeit)
end;

%Ergebnis: dieselbe Technik wie bei Kurvenplots
%geht auch bei den parametrisierten Kurven.
%Wir machen dasselbe nochmal mit einer Epizykloide (laeuft aussen auf einem Kreis)
%Laeuft man innen auf dem Kreis entsteht eine Hypozykloide.
figure(2)
t=linspace(-pi,pi);   rho=1;   r=2;
for a=0.5:0.5:1.5;
  z=r*exp(i*t);
  x=(r+rho)*cos(t)-a*cos((r+rho)*t/rho);
  y=(r+rho)*sin(t)-a*sin((r+rho)*t/rho);
  plot(x,y);hold on;plot(z,'r');hold off
  if a<rho
    xlabel('Gedehnte Epizykloide');end;
  if a==rho
    xlabel('Gemeine Epizykloide');end;
  if a>rho
    xlabel('Verschlungene Epiykloide');end;
  axis equal,pause(Zeit)
end;

figure(3)
t=linspace(-pi,pi);   rho=0.5;   r=2;
for a=[0.25,0.5,1,2]
  z=r*exp(i*t);
  x=(r-rho)*cos(t)+a*cos((r-rho)*t/rho);
  y=(r-rho)*sin(t)-a*sin((r-rho)*t/rho);
  plot(x,y);hold on;plot(z,'r');hold off
  if a<rho
```

```
50      xlabel('Gedehnte Hypozykloide');end;
51    if a==rho
52      xlabel('Gemeine Hypozykloide');end;
53    if a>rho
54      xlabel('Verschlungene Hypozykloide');end;
55    axis equal,pause(Zeit);
56  end;
57  close(1:3); %alle drei Fenster werden geschlossen
58
59  %Wie zeichnet man eine 3D-Kurve, also so etwas wie einen Draht im R^3
60  %Zylindrische Spirale
61  figure(1)
62  t=linspace(0,10*pi,300);
63  x=cos(t);
64  y=sin(t);
65  z=t;
66  plot3(x,y,z);
67  ylabel('Weinwendige Kurve'), pause(Zeit);
68
69  %Zylindrische Spirale anders herum
70  figure(1)
71  t=linspace(0,10*pi,300);
72  x=-cos(t);
73  y=sin(t);
74  z=t;
75  plot3(x,y,z,'r');
76  ylabel('Hopfenwendige Kurve'),pause(Zeit)
77
78  %Bettfeder
79  figure(1)
80  t=linspace(0,10*pi,300);
81  x=(10*pi-t).*cos(t);
82  y=(10*pi-t).*sin(t);
83  z=t;
84  plot3(x,y,z);
85  xlabel('Weinwendige Bettfeder'),pause(Zeit)
86
87  %Bettfeder aus Verschiedenen Perspektiven mit stem-Plots
88  figure(2)
89  lenge=6*pi;
90  t=linspace(0,lenge,300);
91  x=(lenge*pi-t).*cos(t);
92  y=(lenge*pi-t).*sin(t);
93  z=t;
94  stem3(x,y,z,'dr'); %rot und 'diamond'=Rhombus
95  xlabel('Verschiedene Blickwinkel'),
96  view([-65,30]),
97
98  figure(3)
99  stem3(x,y,z,'dr'); %rot und 'diamond'=Rhombus
100 xlabel('Verschiedene Blickwinkel'),
101 view([65,30]),
102
103 figure(4)
104 stem3(x,y,z,'dr'); %rot und 'diamond'=Rhombus
105 xlabel('Verschiedene Blickwinkel'),
106 view([65,-30]),
107
108 figure(5)
109 stem3(x,y,z,'dr'); %rot und 'diamond'=Rhombus
110 xlabel('Verschiedene Blickwinkel'),
111 view([65,100]),
112
113 pause(Zeit);    close(1:5);
114 disp('Start vom Demoprogramm [umfangreich], klick Visualisation');
115 demo
```

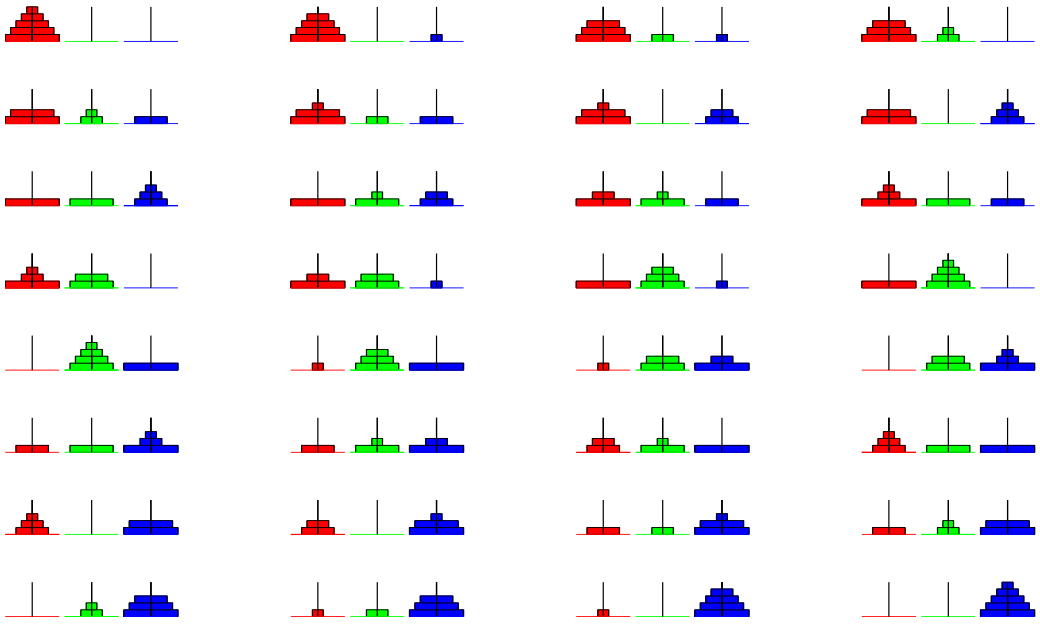
Programm a10_01.m Tonys Trick zur Vervielfachung von Vektoren zu Matrizen

```
%Pascal/MATLAB-Kurs WS 2001, Gerhard Opfer
%Es geht darum, mit moeglichst wenig Rechenaufwand einen Vektor
%als Matrix zu wiederholen. Die jetzt zu besprechende Vervielfachungstechnik
%ist als "Tonys Trick" bekannt.

%Ist x ein gegebener Spaltenvektor, so moechte man eine Matrix herstellen,
%in der jede Spalte x enthaelt, und entsprechend: ist y ein Zeilenvektor,
%so moechte man eine Matrix erzeugen, in der jede Zeile mit y uebereinstimmt. Sei
x=[1:5]'       %ein Spaltenvektor, dann ist
x(:,1)         %derselbe Vektor. Ein Vektor ist also auch eine Matrix.
x(:,[1,1])     %ist derselbe Vektor zweimal nebeneinander geschrieben und
x(:,[1;1])     %liefert dieselbe Verdoppelung. Entsprechend
x(:,ones(6,1)) %wird jetzt x als Spalte versechsfacht. Dasselbe geht auch mit Zeilen.
y=1./[1:6]
y(ones(5,1),:)
y(ones(1,5),:) %liefern beide eine Verfuenffachung von y

%Man kann also leicht die dyadische Summe von x und y  bilden:
%vgl. Programm a05_01.m
X=x(:,ones(length(y),1))
Y=y(ones(1,length(x)),:)
dyadische_Summe=X+Y
%Man braucht genau m*n (hier also 30) Additionen (flops)
```

Programm a10_02.m Turm von Hanoi, rekursive Umschichtung

```
%Pascal/MATLAB-Kurs WS 2001, Gerhard Opfer
%Turm von Hanoi, rekursive Prozedur
%nach einem Pascal-Programm von Elsbeth Bredendiek (ca. 1990).
%Diese Pascal-Prozedur hatte nur die folgenden 9 Zeilen:
%
%  procedure hanoi(n:integer;A,B,C:char);
%  begin
%    if n>0 then
%    begin
%      hanoi(n-1,A,C,B);
%      writeln('Stein ',n:3,' von Turm ',A,' nach Turm ', C);
%      hanoi(n-1,B,A,C);
%    end; {if n}
%  end; {hanoi}
%
%Es geht darum, einen Turm in Position 1, der von oben nach unten n "Steine"
%verschiedener, wachsender Groesse 1,2,...,n enthaelt, unter zuhilfenahme eines
%Turms in Position 2 auf einen Turm in Position 3 so umzuschichten, dass auf
%dem Turm in Position 3 dieselbe Anordnung entsteht wie am Anfang auf dem
%Turm in Position 1 unter Benutzung der folgenden Regeln:
%(a) Es wird jeweils nur ein Stein, naemlich der oberste eines Turms bewegt,
%(b) niemals liegt ein groesserer Stein auf einem kleineren Stein.
%
%Der Inhalt der drei Tuerme wird in jedem Zwischenschritt in den drei
%Vektoren Turm1,Turm2,Turm3 gespeichert. Diese Information
%wird zum Zeichnen der Tuerme benutzt.
%Diese Prozedur wird mit dem Programm "turmcall" aufgerufen. Zur
%Verfuegung stehen muss auch das Programm "turmzeichnen".
%function []=Hanoi(n,Pos1,Pos2,Pos3);

%Diese Datei ist vor Benutzung umzubenennen in Hanoi.m

function []=Hanoi(n,Pos1,Pos2,Pos3);
global Turm1; global Turm2; global Turm3; global z; global mfix;
global Farbe1; global Farbe2; global Farbe3;
if n>0
Hanoi(n-1,Pos1,Pos3,Pos2);
  disp(['Verlagerung von Stein ',int2str(n),' von Pos ',...
  int2str(Pos1),' auf Pos ',int2str(Pos3)]);
```

```
40    s1=['j1=find(Turm',int2str(Pos1),'==n);Turm',int2str(Pos1),'(j1)=[];'];
41    s2=['Turm',int2str(Pos3),'=[Turm',int2str(Pos3),',n];'];
42    eval(s1);   eval(s2);   z=z+1;
43    if mfix==1
44      subplot(1,2,z); else
45      subplot(2^(mfix-2),4,z);
46    end;
47    turmzeichnen(mfix,Turm1,Turm2,Turm3,Farbe1,Farbe2,Farbe3);
48    Hanoi(n-1,Pos2,Pos1,Pos3);   %Es werden insgesamt 2^n-1 Schritte benoetigt.
49  end; %if n>0
50
51  %Ergebnis aus den Pascal-Programm
52  %(dieses MATLAB-Programm liefert dasselbe)
53  %Wie hoch soll der Turm sein? 4
54  %Scheibe  1 von 1 nach 2
55  %Scheibe  2 von 1 nach 3
56  %Scheibe  1 von 2 nach 3
57  %Scheibe  3 von 1 nach 2
58  %Scheibe  1 von 3 nach 1
59  %Scheibe  2 von 3 nach 2
60  %Scheibe  1 von 1 nach 2
61  %Scheibe  4 von 1 nach 3
62  %Scheibe  1 von 2 nach 3
63  %Scheibe  2 von 2 nach 1
64  %Scheibe  1 von 3 nach 1
65  %Scheibe  3 von 2 nach 3
66  %Scheibe  1 von 1 nach 2
67  %Scheibe  2 von 1 nach 3
68  %Scheibe  1 von 2 nach 3  %in 2^n-1=15 Schritten
```



Figur 10.1. Der Turm von Hanoi, alle Schritte bei Turmhöhe $n = 5$

Programm a10_03.m Zeichnen der Türme in den Zwischenpositionen

```
%Hilfsprogramm zum Turm von Hanoi
%Drei Tuerme mit maximal m Kreisscheiben (gen. Steine) werden gezeichnet.
%Die Groesse und Anzahl der Kreisscheiben ist in den Vektoren
%Turm1, Turm2, Turm3 gespeichert.
%Turm1(1) enthaelt den groessten, untersten Stein von Turm1, etc.

%Diese Datei ist vor Benutzung umzubenennen in turmzeichnen.m

function []=turmzeichnen(m,Turm1,Turm2,Turm3,Farbe1,Farbe2,Farbe3);
global mfix;
for Fall=1:3
  t=['Turm=Turm',int2str(Fall),';'];
  f=['Farbe=Farbe',int2str(Fall),';'];
  eval(t); eval(f);
  if Fall==1, Pos=-2*m-1;end;
  if Fall==2, Pos=0;end;
  if Fall==3, Pos=2*m+1;end;
  l=length(Turm);
  if l==0                              %wenn der Turm leer ist
    plot([-m+Pos,m+Pos],[0,0],Farbe); %Basislinie
    if Fall==1, hold on; end; %if Fall
    plot([Pos,Pos],[0,m],'k');         %senkrechte Mittellinie
    end; %if l
  for j=1:l
    if Fall==1, hold on; end; %if
    k=Turm(j);
    plot([Pos+k,Pos-k],[j-1,j-1],Farbe,...
    [Pos-k,Pos-k],[j-1,j],Farbe,...
    [Pos-k,Pos+k],[j,j],Farbe,...
    [Pos+k,Pos+k],[j,j-1],Farbe);
    fill([Pos-k,Pos-k,Pos+k,Pos+k],[j-1,j,j,j-1],Farbe);
    plot([Pos,Pos],[0,m],'k');
    plot([-m+Pos,m+Pos],[0,0],Farbe);
  end; %for j
  if Fall==3 hold off, axis([-4*m,4*m,-2,m+2]), axis off, end; %if
end; %for Fall
```

Programm a10_04.m Aufrufprogramm zum Turm von Hanoi

```
%Aufrufprogramm zum Turm von Hanoi. Zur Verfuegung stehen muss auch das
%Programm "turmzeichnen". Es werden alle 2^n Zwischenpositionen in kleinen
%subplots auf ein Blatt gezeichnet. Das funktioniert bei moderatem m<=6.
global Turm1; global Turm2; global Turm3; global z; global mfix;
global Farbe1; global Farbe2; global Farbe3;
z=0;
m=5; mfix=m;
m1=1;m2=2;m3=3;
Farbe1='r';Farbe2='g';Farbe3='b';
Turm1=[m:-1:1]; Turm2=[]; Turm3=[];
z=z+1;
if m==1
  subplot(1,2,z), else
  subplot(2^(m-2),4,z);
end;
turmzeichnen(mfix,Turm1,Turm2,Turm3,Farbe1,Farbe2,Farbe3);
s=['Anfang mit n = ',int2str(m),' ========================='];
disp(s);
Hanoi(m,m1,m2,m3);
disp('Ende=======================================');
```

Programm a11_01.p Variable vom Typ set am Beispiel eines Programms zur Lottoziehung

```
 1 {Pascal/MATLAB-Kurs Oktober 2000, Gerhard Opfer
 2 aus dem Arsenal von Elsbeth Bredendiek}
 3 program Definition_und_Gebrauch_von_Mengen;
 4 {Simulation einer Lotto-Veranstaltung.
 5 1. 20 Tips werden abgegeben, Tip = 6 verschiedene Zahlen in 1 bis 49
 6 2. Ziehung wird durchgefuehrt,
 7 3. Tips werden ausgewertet.}
 8 type Lottotip=set of 1..49;
 9 var i,j,anz,Wieviel_Tips:integer;
10     Ziehung,Tip,Richtige:Lottotip;
11     Trefferzahl:array[0..6] of integer;
12 procedure Auswahl(var Menge:Lottotip);
13 var i,zuf,ohne_Bedeutung:integer;
14 begin
15   Menge:=[]; {leere Menge}
16   for i:=1 to 6 do
17   begin
18     repeat   {random in [0,1] ===> 48*random+1 in [1,49]}
19       {in DOS random(49)}
20       zuf:=round(48*random(ohne_Bedeutung)+1); {in DOS ohne_Bedeutung weglassen}
21       {zuf:=round(48*random+1);} {DOS}
22     until not (zuf in Menge); {damit alle 6 Zahlen verschieden sind}
23     Menge:=Menge+[zuf]; {Mengenoperation Vereinigung}
24     write(zuf:3);
25   end; {for}
26 end; {Auswahl}
27 begin {Hauptprogramm}
28   for i:=0 to 6 do Trefferzahl[i]:=0;
29   write('Wieviele Tips? ');
30   readln(Wieviel_Tips);
31   writeln('Ziehung');
32   writeln;
33   Auswahl(Ziehung);
34   writeln;writeln;
35   writeln('Verschiedene Tips:');
36   writeln;
37   for i:=1 to Wieviel_Tips do
38   begin
39     Auswahl(Tip);
40     Richtige:=Ziehung*Tip; {Mengenoperation Durchschnitt}
41     {wieviele Richtige?}
42     anz:=0;
43     j:=1;
44     while Richtige<>[] do
45     begin
46       if j in Richtige then
47       begin
48         anz:=anz+1;
49         Richtige:=Richtige-[j]; {Mengenoperation Komplement:}
50                                {nicht j aber in Richtige}
51       end; {if j }
52       j:=j+1;
53     end; {while}
54     writeln(anz:3,' Richtige');
55     Trefferzahl[anz]:=Trefferzahl[anz]+1;
56   end; {for i}
57   writeln;
58   writeln('Null Richtige: ',Trefferzahl[0]:4);
59   writeln('Ein Richtiger: ',Trefferzahl[1]:4);
60   writeln('Zwei Richtige: ',Trefferzahl[2]:4);
61   writeln('Drei Richtige: ',Trefferzahl[3]:4);
62   writeln('Vier Richtige: ',Trefferzahl[4]:4);
63   writeln('Fuenf Richtige: ',Trefferzahl[5]:4);
```

```
  writeln('Sechs Richtige:',Trefferzahl[6]:4);
  writeln;
end.

{Ergebnisse:
Wieviele Tips? 20
Ziehung

 26  9 16 27 46 35

Verschiedene Tips:

 12 25  7  5 20 14  0 Richtige
 19 48 27 38 32 41  1 Richtige
  8 31 16 18 45 26  2 Richtige
 20 30 39 46 43 33  1 Richtige
 37 29 20 18 11 41  0 Richtige
 21 23 48  7 11 47  0 Richtige
 36 21 38 37 47  2  0 Richtige
 16 37 13 29  3 47  1 Richtige
 16  4 22 45 28  7  1 Richtige
 28 13 25 12 24 20  0 Richtige
 43 21 18 19  3  9  1 Richtige
 26 34  6 20 38 13  1 Richtige
 17 12 15 16 44  3  1 Richtige
 32 20 33 36 46 12  1 Richtige
 41 47 38 22 33 40  0 Richtige
  9 14  7 42 37 11  1 Richtige
  8 15 40 12 28 35  1 Richtige
 10 49 13 22 37 42  0 Richtige
 44 48 20 22  7 23  0 Richtige
 12 48 32 30 13 23  0 Richtige

Null Richtige:    9
Ein Richtiger:   10
Zwei Richtige:    1
Drei Richtige:    0
Vier Richtige:    0
Fuenf Richtige:   0
Sechs Richtige:   0}
```

Programm a11_02.p Ein Programm mit Variablen vom Typ `record`. Das sind Variablen mit denen man z. B. den (heterogenen) Inhalt von Karteikarten simulieren kann.

```
{Pascal/MATLAB-Kurs Oktober 2000, Gerhard Opfer}
program records;
{Wir stellen einen neuen Variablentyp 'records' vor, den man vielleicht
als Inhalt einer Karteikarte interpretieren kann.
Wir geben einige einfache Beispiele.}
{Funktionsprozeduren mit Wert vom record-Typ werden von Borland-Pascal
nicht zugelassen. Funktioniert aber gut auf Sun-Pascal.}
{------------------------------------------------------------------------}
type T=1..31; {Ordinaltyp, Aufzaehlungstyp}
     M=(Januar,Februar,Maerz,April,Mai,Juni,Juli,August,
                     September,Oktober,November,Dezember);
     J=1900..2100;

     Datum=record Tag:T;
                  Monat:M;
                  Jahr:J;
                  end;

     komplexe_zahl=record Realteil:real;
                          Imaginaerteil:real;
                          end;

```

```
23 var z,z1,z2:komplexe_zahl;
24     D,E:Datum;
25 {-------------------------------------------------------------------------------}
26 {*}function Addition(u,v:komplexe_zahl):komplexe_zahl;
27 {*}begin
28 {*}  Addition.Realteil:=u.Realteil+v.Realteil;
29 {*}  Addition.Imaginaerteil:=u.Imaginaerteil+v.Imaginaerteil;
30 {*}end; {Addition}
31 {-------------------------------------------------------------------------------}
32 {*}function Multiplikation(u,v:komplexe_zahl):komplexe_zahl;
33 {*}var x:real;
34 {*}begin
35 {*}x:=(u.Realteil+u.Imaginaerteil)*v.Realteil;
36 {*}Multiplikation.Imaginaerteil:=x-u.Realteil*(v.Realteil-v.Imaginaerteil);
37 {*}Multiplikation.Realteil:=x-u.Imaginaerteil*(v.Realteil+v.Imaginaerteil);
38 {*}{Es werden nur drei Multiplikationen gebraucht, statt standardmaessig vier}
39 {*}end; {Multiplikation}
40 {-------------------------------------------------------------------------------}
41 {*}function naechster_Tag(d:Datum):Datum;
42 {*}{Ausgehend vom Datum d, soll das Datum des naechsten Tages bestimmt werden}
43 {*}var t:T;
44 {*}    m:M;
45 {*}    j:J;
46 {-------------------------------------------------------------------------------}
47 {**}function letzter_Tag(d:Datum):boolean;
48 {**}(*Es soll festgestellt werden, ob das gegebene Datum auf den letzten
49 {**}Tag eines Monats faellt. Es wird unterstellt, dass das Datum richtig ist,
50 {**}also dass z. B. der 29. Februar 2001 oder der 31. April nicht vorkommt*)
51 {-------------------------------------------------------------------------------}
52 {***}function letzter_Tag_Februar(jj:J):T;
53 {***}{Es wird der letzte Tag vom Februar im Jahr jj ausgerechnet}
54 {***}begin  {letzter_Tag_Februar}
55 {***}if ((jj mod 4=0) and (jj mod 100 <> 0)) or (jj mod 400=0) then
56 {***}  letzter_Tag_Februar:=29 else
57 {***}  letzter_Tag_Februar:=28;
58 {***}end;{letzter_Tag_Februar}
59 {-------------------------------------------------------------------------------}
60 {**}begin {letzter_Tag}
61 {**}t:=d.Tag;
62 {**}m:=d.Monat;
63 {**}j:=d.Jahr;
64 {**}if (((m=Januar) or (m=Maerz) or (m=Mai) or (m=Juli) or (m=August)
65 {**}   or (m=Oktober) or (m=Dezember)) and (t=31)) or
66 {**}   (((m=April) or (m=Juni) or (m=September) or (m=November)) and (t=30)) or
67 {**}   ((m=Februar) and (t=letzter_Tag_Februar(j))) then
68 {**}   letzter_Tag:=true
69 {**}   else
70 {**}   letzter_Tag:=false;
71 {**}end; {letzter_Tag}
72 {-------------------------------------------------------------------------------}
73 {*}begin {naechster_Tag}
74 {*}t:=d.Tag;
75 {*}m:=d.Monat;
76 {*}j:=d.Jahr;
77 {*}if letzter_Tag(d) then
78 {*}begin
79 {*}  naechster_Tag.Tag:=1;
80 {*}  if m<>Dezember then
81 {*}  begin
82 {*}    naechster_Tag.Monat:=succ(m);
83 {*}    naechster_Tag.Jahr:=j;
84 {*}  end else {also Dezember}
85 {*}  begin
86 {*}    if j<>2100 then
87 {*}    begin
88 {*}      naechster_Tag.Monat:=Januar;
```

```
 89 {*}      naechster_Tag.Jahr:=succ(j);
 90 {*}    end else {also j=2100}
 91 {*}    begin
 92 {*}      naechster_Tag:=d;
 93 {*}      writeln(
 94 {*}      'obere Jahresgrenze erreicht, Nachfolgedatum 1. Januar 2101');
 95 {*}      writeln(' kann nicht mehr bestimmt werden:');
 96 {*}    end;
 97 {*}  end;
 98 {*}end else {also nicht letzter Tag}
 99 {*}begin
100 {*}    naechster_Tag.Tag:=succ(t);
101 {*}    naechster_Tag.Monat:=m;
102 {*}    naechster_Tag.Jahr:=j;
103 {*}end;
104 {*}end; {naechster_Tag}
105 {------------------------------------------------------------------------}
106 begin {Hauptprogramm}
107 {Zuerst Tests mit Typ komplexe_zahl-------------------------------------}
108   z1.Realteil:=3;
109   z1.Imaginaerteil:=4;
110   z2.Realteil:=-1;
111   z2.Imaginaerteil:=2;
112   writeln(z1.Realteil:4:0,z1.Imaginaerteil:4:0);
113   if z1=z2 then writeln('Vergleich von records geht: gleich')
114      else writeln('Vergleich von records geht: ungleich');
115   z:=Addition(z1,z2);
116   writeln(z.Realteil:4:0,z.Imaginaerteil:4:0);
117   z:=Multiplikation(z1,z2);
118   writeln(z.Realteil:4:0,z.Imaginaerteil:4:0);{(*)}
119   {Kommt ein record-Name oft vor, kann man ihn weglassen,
120   muss aber vorher 'with Name do' schreiben:}
121   with z do  {ggf begin ... end;}
122   writeln(Realteil:4:0,Imaginaerteil:4:0);{Dasselbe wie (*)}
123 {Jetzt Tests mit Typ Datum---------------------------------------------}
124   D.Tag:=28;
125   D.Monat:=Februar;
126   D.Jahr:=2000;
127   E.Tag:=28;
128   E.Monat:=Februar;
129   E.Jahr:=2000;
130   if D=E then writeln('Vergleich von records geht: gleich')
131      else writeln('Vergleich von records geht: ungleich');
132
133   D.Tag:=31; D.Monat:=Maerz; D.Jahr:=2100;
134   E:=naechster_Tag(D);
135   write(D.Tag:3,'.',D.Monat:10,D.Jahr:5, ' naechster Tag: ');
136   writeln(E.Tag:3,'.',E.Monat:10,E.Jahr:5);
137
138   D.Tag:=29; D.Monat:=Juli; D.Jahr:=2001;
139   E:=naechster_Tag(D);
140   write(D.Tag:3,'.',D.Monat:10,D.Jahr:5, ' naechster Tag: ');
141   writeln(E.Tag:3,'.',E.Monat:10,E.Jahr:5);
142
143   D.Tag:=28; D.Monat:=Februar; D.Jahr:=2000;
144   E:=naechster_Tag(D);
145   write(D.Tag:3,'.',D.Monat:10,D.Jahr:5, ' naechster Tag: ');
146   writeln(E.Tag:3,'.',E.Monat:10,E.Jahr:5);
147
148   D.Tag:=28; D.Monat:=Februar; D.Jahr:=1999;
149   E:=naechster_Tag(D);
150   write(D.Tag:3,'.',D.Monat:10,D.Jahr:5, ' naechster Tag: ');
151   writeln(E.Tag:3,'.',E.Monat:10,E.Jahr:5);
152
153   D.Tag:=31; D.Monat:=Dezember; D.Jahr:=2001;
154   E:=naechster_Tag(D);
```

```
155   write(D.Tag:3,'.',D.Monat:10,D.Jahr:5, ' naechster Tag: ');
156   writeln(E.Tag:3,'.',E.Monat:10,E.Jahr:5);
157
158   D.Tag:=31; D.Monat:=Dezember; D.Jahr:=2100;
159   E:=naechster_Tag(D);
160   write(D.Tag:3,'.',D.Monat:10,D.Jahr:5, ' naechster Tag: ');
161   writeln(E.Tag:3,'.',E.Monat:10,E.Jahr:5);
162 end. {Hauptprogramm}
163 {-------------------------------------------------------------------------------}
164 {Ergebnisse:
165    3   4
166 Vergleich von records geht: ungleich
167    2   6
168  -11   2
169  -11   2
170 Vergleich von records geht: gleich
171  31.     Maerz 2100 naechster Tag:   1.     April 2100
172  29.      Juli 2001 naechster Tag:  30.      Juli 2001
173  28.   Februar 2000 naechster Tag:  29.   Februar 2000
174  28.   Februar 1999 naechster Tag:   1.     Maerz 1999
175  31.  Dezember 2001 naechster Tag:   1.    Januar 2002
176 obere Jahresgrenze erreicht, Nachfolgedatum 1. Januar 2101
177  kann nicht mehr bestimmt werden:
178  31.  Dezember 2100 naechster Tag:  31.  Dezember 2100}
```

ENDE