Begleitblätter zu den Übungen **Grundlagen verfahrenstechnischer Simulation** und **Informatikanwendungen für Wirtschaftsingenieure**
TFH Berlin, FB VIII

Elemente des Programmierens, dargestellt an Hand des Computeralgebrasystems MAPLE sowie der Programmierumgebung VISUAL BASIC

Gegenstand	**Darstellung** in MAPLE	in BASIC
Abschluß einer Anweisung:	; (wenn Ausgabe gewünscht) : (wenn Ausgabe nicht gewünscht)	Zeilenwechsel oder :
Kennzeichnung eines **Kommentars**:	#	‘
Namen von Variablen:	– Dürfen nur Buchstaben, Ziffern und Unterstriche (_) enthalten – Müssen mit einem Buchstaben beginnen – Dürfen nicht mit einem Schlüsselwort (z.B. `if, for`) übereinstimmen	
Zeichenketten (strings): (Dürfen beliebige Zeichen enthalten)	`"zeichenfolge"`	
Verknüpfung:	`cat("folge1","folge2")`	`"folge1"&"folge2"` (auch: `"folge1"+"folge2"`) Bis MAPLE V: ‘`folge‘` statt `"folge"` (Leertaste nach ‘)

Einige zulässige **Dezimaldarstellungen** der Zahlen 1870 bzw. 1/25:

`1870.0`	`1870.`	`.187*10^4`	`.187E4`	`1.87e3`
`0.04`	`.04`	`4.*10^(-2)`	`.4E-1`	`4.0e-2`

Arithmetische Operationen mit Zahlen oder mit Variablen, deren Werte Zahlen sind:

Addition	$a+b$		`a+b`
Subtraktion	$a-b$		`a-b`
Multiplikation	ab		`a*b`
Division	$\frac{a}{b}$		`a/b`
Ganzzahlige Division (n, m, p, q ganzzahlig)	$\frac{n}{m} = p$ Rest q	`p:=iquo(n,m)` `q := n mod m`	`p=n\m`
Potenzieren	a^b		`a^b` (Leertaste nach `^`)
Klammern	$a(b+c)$		`a*(b+c)`

Eingebaute Funktionen:

Betrag	$\|x\|$		`abs(x)`	
Vorzeichen	signum(x)	`signum(x)`		`sgn(x)`
Sprungfunktion	$\sigma(x)$	`Heaviside(x)`		`(1+sgn(x))/2`
Ganzzahliger Anteil		`trunc(x)`		`fix(x)`
Gebrochener Anteil		`frac(x)`		`x-fix(x)`
Quadratwurzel	$\sqrt{x}$	`sqrt(x)`		`sqr(x)`
n-te Wurzel	$\sqrt[n]{x}$ (n ganz)	`surd(x,n)`		`x^(1/n)`
Exponentialfunktion zur Basis **e**	$\mathbf{e}^x$		`exp(x)`	
Natürlicher Logarithmus	ln x		`log(x)`	
		`ln(x)`		
Sinus	sin x		`sin(x)`	
Cosinus	cos x		`cos(x)`	
Tangens	tan x		`tan(x)`	
Hyperbelfunktionen				
	sinh x	`sinh(x)`		`(exp(x)-exp(-x))/2`
	cosh x	`cosh(x)`		`(exp(x)+exp(-x))/2`
	tanh x	`tanh(x)`		`(exp(2*x)-1)/(exp(2*x)+1)`
Arcus-Funktionen				
	arcsin x	`arcsin(x)`		
	arccos x	`arccos(x)`		
	arctan $\frac{y}{x}$	`arctan(y/x)`		`atn(y/x)`
		`arctan(y,x)`		
Area-Funktionen				
	arsinh x	`arcsinh(x)`		
	arcosh x	`arccosh(x)`		
	artanh x	`arctanh(x)`		

Werte von **Booleschen Variablen**:

wahr	`true`
falsch	`false`

Logische Operationen:

und (beide Bedingungen wahr)	$a \wedge b$	`and`
oder (mindestens eine wahr)	$a \vee b$	`or`
entweder .. oder (genau eine wahr)	$(a \vee b) \wedge \neg(a \wedge b)$	`xor`
nicht	$\neg a$	`not`

Vergleichsoperatoren:

kleiner als	$<$	`<`
kleiner oder gleich	$\leq$	`<=`
größer als	$>$	`>`
größer oder gleich	$\geq$	`>=`
gleich	$=$	`=`
ungleich	$\neq$	`<>`

Zuordnung eines Wertes zu einer Variablen: `:=` `=` (Vorsicht, Doppelbedeutung des Gleichheitszeichens!)

Zählschleifen:

Durchlauf für alle Werte der Zählvariablen i vom Anfangswert j bis zum Endwert k mit Schrittweite m

```
[for i from j by m]
to k do
  Anweisungen
end do;
```

```
for i=j to k [step m]
  Anweisungen
next [ i ]
```

[Das durch eckige Klammern Gekennzeichnete darf fehlen]

Angabe der Zählvariablen als Liste

```
for i in liste do
  Anweisungen
end do;
```

Bedingte Schleifen:

Kopfgesteuert
(Durchlauf so lange, wie eine Bedingung erfüllt ist)

```
while Bedingung
do Anweisungen
end do;
```

```
do while Bedingung
  Anweisungen
loop
```

Fußgesteuert
(Durchlauf so lange, wie eine Bedingung erfüllt ist — mindestens jedoch einmal)

```
do Anweisungen
if not Bedingung
then break end if
end do;
```

```
do
  Anweisungen
loop while Bedingung
```

Variante in BASIC:
Statt `while` *Bedingung* kann man `until not` *Bedingung* verwenden.
(z.B. `while` i<>0 $\Longleftrightarrow$ `until` i=0)

Bis MAPLE V muss, ab MAPLE 6 kann `end do` durch `od` abgekürzt werden

Verzweigungen:

Die drei wichtigsten Arten
(dargestellt für BASIC)

`if` *Bedingung* `then`
  Anweisungen
`end if`

`if` *Bedingung* `then`
  Anweisungen1
`else`
  Anweisungen2
`end if`

`if` *Bedingung1* `then`
  Anweisungen1
`elseif` *Bedingung2* `then`
  Anweisungen2
`else`
  Anweisungen3
`end if`

Abweichungen in MAPLE:
Zeilenwechsel ohne Bedeutung
`elseif` → `elif`
Bis MAPLE V muss, ab MAPLE 6 kann
`end if` durch `fi` abgekürzt werden

Möglichkeit nur in BASIC:
`if` *Bedingung* `then` *Anweisung*

Prozeduren, Funktionen, Unterprogramme:
(procedure, function, subroutine)

[Das im folgenden durch eckige Klammern Gekennzeichnete darf fehlen]

MAPLE:

Definition einer **Prozedur**: *Name*:=`proc`(*formale Parameter*[::*Typ*])
[*Deklarationen*]
[*Anweisungen*]
`end proc;`

Ausgegeben wird der Wert der letzten Anweisung
Bis MAPLE V muss, ab MAPLE 6 kann `end proc` durch `end` abgekürzt werden

Aufruf der Prozedur: *Name*(*aktuelle Parameter*);

BASIC:

Definition einer **Funktion**:

`function` *Name*(*formale Parameter*)
[*Deklarationen*]
[*Anweisungen*]
Name = Wert
`end function`

Aufruf der Funktion: *Variable = Name(aktuelle Parameter)*

Definition eines **Unterprogramms**:

`sub` *Name*(*formale Parameter*[`as` *Typ*])
[*Deklarationen*]
Anweisungen
`end sub`

Aufruf des Unterprogramms: *Name aktuelle Parameter*
oder: `call` *Name(aktuelle Parameter)*

Besonderheiten von MAPLE:

Wichtige **Strukturen**:

sequence (Sequenz)	`a,b,c`
Leere Sequenz	`NULL`
list (Liste)	`[a,b,c]`
Leere Liste	`[ ]`
listlist (Liste von Listen)	`[[a,b],[c,d]]`
set (Menge)	{a,b,c}
Leere Menge	{ }
Im Rahmen des Pakets `linalg` ferner:	
vector (Spaltenmatrix, Vektor)	`vector([a,b,c])`
matrix (Matrix)	`matrix([[a,b,c],[d,e,f],[g,h,i]])`

Konstanten:

unendlich ∞	infinity
π	Pi
e	exp(1)
i	I

Wichtige **Pakete**(package):

linalg	Lineare Algebra (lineare Gleichungen, Vektoren, Matrizen)
plots	Spezielle Graphikmöglichkeiten
plottools	Geometrische Objekte
stats	Statistik und Wahrscheinlichkeitsrechnung
simplex	Simplexverfahren

Einlesen z.B. `with(plots):`

Zurücksetzen auf den Ausgangszustand:

```
restart:
```

Hilfethema aufrufen, z.B.

```
?plot,options
```

Wichtige **MAPLE-Befehle**:		Einige Beispiele:
Die zwei Arten von Anweisungen:		
Zuordnungsanweisung		`a:=3.5;`
Abfrageanweisung		`a;`
Auswertung:		
assign	Wert zuweisen	`gl:=a=3.5; assign(gl):`
	Wert entziehen	`a:='a':`
subs	Einsetzen	`y:=subs(x=Pi,cos(x));`
eval	Auswerten (z.B. nach subs)	`eval(y);`
evalf	Auswerten als Dezimalzahl	`evalf(sqrt(2));`
evalc	Zerlegen einer komplexen Zahl	`evalc(exp(3*I));`
Vektor- und Matrizenrechnung:		
evalm	Anwenden der Regeln der Matrizenrechnung	`evalm(a+b);`
det	Determinante einer Matrix	`det(A);`
inverse	Kehrmatrix	`inverse(A);`
`&*`	Multiplikation von Matrix und Vektor	`evalm(A&*b);`
innerprod	Punktprodukt, Skalarprodukt	`innerprod(a,b);`
crossprod	Kreuzprodukt, Vektorprodukt	`crossprod(a,b);`
norm(,2)	Betrag eines Vektors	`norm(a,2);`
	Komponente eines Vektors (oder einer Liste)	`a[7];`
	Komponente einer Matrix (oder einer Liste von Listen)	`A[7,4];`
Sequenzen, Summen, Produkte:		
seq	Sequenz erzeugen	`seq(i^2-5*i,i=1..10);`
		`s:= seq(i^2-5*i,i=[2,7,15]);`
min	Minimum einer Sequenz	`min(s);`
add	Sequenz addieren	`add(i^2-5*i,i=1..10);`
mul	Sequenz multiplizieren	`mul(i^2-5*i,i=1..10);`
sum	Summe, endlich oder unendlich	`sum(1/i^2,i=1..infinity);`
Gleichungslösung:		
lhs, rhs	linke, rechte Seite einer Gleichung	`lhs(x^2=y);`
solve	Gleichungslösung	`solve(x^5+x^2-5=0,x);`
RootOf	Menge der Lösungen einer Gleichung	
fsolve	Gleichungslösung numerisch	`fsolve(x^5+x^2-5=0,x);`
linsolve	Lösen einer linearen Matrizengleichung	`linsolve(A,b);`
dsolve	Lösen einer Differentialgleichung	

Analysis:

limit	Grenzwert	`limit(sin(x)/x,x=0);`
diff	Ableiten	`diff(sin(a*x),x);`
int	Integrieren	`int(sin(a*x),x);`
evalf(Int( ))	Numerische Integration	`evalf(Int(sqrt(cos(x^3)),x=0..1));`
	Ab MAPLE 6 genügt:	`int(sqrt(cos(x^3)),x=0..1.0);`
series	Reihenentwicklung	`series(sin(a*x),x=0);`

Graphik:

plot	Zweidimensionale Graphik erstellen	
	von einem Funktionsausdruck	`plot(x^5,x);`
	von einer Prozedur (z.B. einer eingebauten Funktion)	`plot(sin);`
	von einem Polygon	`plot([[0,1],[1,1],[2,3]]);`
plot3d	Dreidimensionale Graphik erstellen	
logplot	Logarithmische Graphik	
loglogplot	Doppeltlogarithmische Graphik	
odeplot	Graphik der numerischen Lösung einer Differentialgleichung	

Formelmanipulation:

select	Auswählen	`select(x->x>0,liste);`
simplify	Vereinfachen	`simplify(sin(x)^2+cos(x)^2);`
expand	Entwickeln	`e:=expand(sin(a+b));`
combine	Zusammenfassen	`combine(e,trig);`
collect	Sammeln	`collect(ausdruck,[x,y]);`
sort	Ordnen (Potenzen, Listen)	`sort(x^2-x^5+x^3);`
convert	In eine andere Form bringen	`convert(tan(x/2),sincos);`
nops	Zahl der Operanden	`nops(liste);`
map	Anwendung einer Funktion auf alle Operanden	`map(x->x^2,liste);`

Fehlersuche:

debug	Ausführliche Ausgabe einer Prozedur einschalten	`debug(name);`
undebug	dito abschalten	`undebug(name);`

Aus- und Eingabe:

print	Ausgabe des Wertes einer Variablen (erforderlich, falls der Wert der Variablen ein Vektor, eine Matrix oder eine Prozedur ist, sonst genügt der Aufruf des Namens)	`print(a);`
read	Einlesen einer externen Datei in die Sitzung	

Zusammengestellt von Prof. Dr.-Ing. A. Krawietz