```
================================================================
"PROGMATH" - Programmieren mit Mathematik (c) Herbert Paukert.
(Skripten-Bibliothek <Bibliothek e/a> mit dem Passwort "free")
================================================================
```



```
================================================================
Dieser Hilfetext kann mittels <Datei-Drucken> gedruckt werden.
================================================================

Das interaktive Programm "PROGMATH" hat zwei Zielvorgaben:
>> Erstens sollen damit Mathematik-Inhalte trainiert werden.
>> Zweitens können mithilfe der Skriptsprache die Grundlagen
   des Programmierens erlernt werden.

"PROGMATH" ist ein Texteditor mit dessen Hilfe so genannte
Mathematik-Skripts erstellt, ausgeführt und gespeichert werden.
Im Texteditor ist dazu ein Mathematik-Interpreter eingebaut.
"PROGMATH" ist in den vier Programmen "GEOMATH" und "MATHEPRO",
"GAPMEDIA" und "MEDIAPRO" integriert.

<Datei-Neu> löscht den Text im Editor.
<Datei-Öffnen> öffnet eine Textdatei (Skriptdatei).
<Datei-Speichern> speichert den aktuellen Text.
<Datei-Drucken> druckt einen Text.
<Datei-Drucken-G> druckt eine Grafik.
<Datei-Editor-E/A> Wechsel zwischen sichtbarem und unsichtbarem
                   Texteditor bei der Skriptausführung.
<Datei-Exit> schließt den Editor.

<Bearbeiten-Schrift> wählt eine Schriftart aus.
<Bearbeiten-Markieren> markiert den gesamten Text.
<Bearbeiten-Suchen> sucht einen Text.
<Bearbeiten-Ersetzen> ersetzt einen Text.
<Bearbeiten-Linker Rand> verschiebt den linken Textrand.
<Bearbeiten-Komprimieren> komprimiert den Text.
<Bearbeiten-Dezimalen> setzt die Anzahl der Dezimalstellen.

<Parse> oder <F1> führt eine Mathematik-Zeile im Editor aus.
<Skript> oder <F11> führt ein Mathematik-Skript im Editor aus.
Dazu muss der Cursor auf den Befehl "begin" platziert werden.
Jedes Skript kann jederzeit mit <Esc> abgebrochen werden.

<Demos-1> öffnet eine Sammlung von 20 einfachen Skripts.
<Demos-2> öffnet eine Sammlung von 20 komplexen Skripts.
<Hilfe e/a> schaltet die Hilfetext-Anzeige an oder aus.

"Drag and Drop" verschiebt markierte Textteile.
<Strg C> kopiert den markierten Text in die Zwischenlage.
<Strg V> fügt den kopierten Text an der Cursorstelle ein.
<Strg><Pos1> springt immer an den Textanfang.
<Strg><Ende> springt immer an das Textende.
Ein Zeilenvorschub erfolgt mittels Taste <Enter>.
Ein Seitenvorschub erfolgt mittels <Strg><Enter>.
Das Zeichen "PI" symbolisiert einen Seitenvorschub.

Der Schalter <Grafik> blendet eine Grafik ein und aus.
Ein rechter Mausklick in die Grafik zeigt die Koordinaten.
```

Hinweis: Bei entsprechender Verwendung der PROGMATH-Befehle können in einem Skript interaktiv Antworten abgefragt und eingegeben und schließlich bewertet werden. Dafür wird der INVAR-Befehl verwendet (siehe Seite 09). Ab Seite 26 befinden sich dazu einige Demoskripts. In den Eingabezellen des INVAR-Befehls kann mit einem Doppelklick ein algebraischer Taschenrechner aufgerufen werden. Beim Schließen des Taschenrechners werden alle Rechenergebnisse automatisch in die Eingabezellen übertragen.

Belegungen der Funktionstasten:
<F1> ... Mathematik-Operation aus einer Zeile durchführen.
<F11> ... Mathematik-Skript aus mehreren Zeilen durchführen.
          <F11> entspricht dem Menü-Eintrag <Skript ausführen>.
<F2> ... alle Zeichencodes der aktuellen Schrift anzeigen.
<F3> ... schaltet die Grafik ein oder aus.
<F4> ... eine sichtbare Grafik drucken oder speichern,
          (hängt davon ab, ob mit <F6> eine Umleitung erfolgte).
<F5> ... eine gespeicherte Grafik laden.
<F6> ... Umleitung von <F4> "Grafik drucken" auf "Grafik speichern".
          (Wechselschalter)
<F7> ... den Taschenrechner einblenden.
<F8> ... unbelegt.
<F9> ... statistische Auswertung von Daten, die untereinander stehen
          oder durch Kommas getrennt nebeneinander stehen. Sie müssen
          mit der Maus genau markiert werden. Dezimalzahlen werden mit
          einem Dezimalpunkt geschrieben.

<F10>   ... Schrift in Grafiken normalisieren.
<Shift><F10> ... Schrift in Grafiken vergrößern.
<Strg><F10>   ... Schrift in Grafiken verkleinern.
<F12>   ... Fettschrift in Grafiken ein-/ausschalten.

<Strg><F7>, <Shift><F7>, <Strg><F8>, <Shift><F8>, <Strg><F9>
und <Shift><F9> sind mit internen Spezialfunktionen belegt.
Diese Funktionen sind nur im sichtbaren Editor ausführbar.

(1) Der PROGMATH-Parser

Es kann in einer Textzeile eine mathematische Rechenformel mit beliebigen Zahlen und den konventionellen Operatoren und Funktionen eingegeben werden. Außerdem können die 26 Variablen a,b,c,... y,z zur Speicherung verwendet werden. Eine mathematische Berechnung wird ausgeführt, indem man den Cursor auf die Zeile stellt und auf <F1> drückt oder auf <Parse> klickt. Dabei bestehen im Mathematik-Parser folgende Möglichkeiten:

a = Zahl	<F1>	speichert die Zahl auf a.
a = Formel(a,b,...)	<F1>	wertet die Formel aus und speichert den Wert auf a.
a =	<F1>	zeigt den Wert von a an.
Formel(a,b,...) =	<F1>	wertet die Formel aus und zeigt den Formelwert an.

OPERATOREN: (,),+,-,*,/,^.
FUNKTIONEN: abs,acos,asin,atan,cos,deg,exp,fak,log,ln,
            rad,round,sin,sqr,sqrt,tan,trunc,pi.

Hinweis: Dezimalzahlen immer mit Dezimalpunkt schreiben.
Die Anzahl der Dezimalstellen ist standardmäßig 2. Sie
kann mit <Bearbeiten-Dezimalstellen> eingestellt werden.

Beginnt eine Zeile mit //, dann wird sie als Kommentarzeile interpretiert und nicht ausgeführt.

Ein Punkt "." am Anfang einer Zeile wird immer ignoriert. Wird ein Punkt vor Wertzuweisungen (.a =) geschrieben, dann können diese Zeilen mit <Bearbeiten-Komprimieren> oder dem Befehl "COMPRESS" nicht entfernt werden. Dieser Befehl ist notwendig um variable Wertzuweisungen aus den Textzeilen zu entfernen, aber fixe Wertangaben zu erhalten.

Neben den einfachen Wertzuweisungen (siehe oben) stehen 140 arithmetische und geometrische Spezialbefehle zur Verfügung, welche unten aufgelistet sind. In den Funktionsformeln F(x) ist x das Argument und es können auch die anderen Variablen a,b,... y,z für Zahlen verwendet werden, z.B. sqrt(a^2-x^2).

In vielen geometrischen Befehlen sind die Parameter in Klammern symbolische Bezeichner für Punkte A,B,C,... Y,Z. Die Punkte müssen vorher gespeichert (gezeichnet) werden, was durch Niederschreiben der Koordinaten und Drücken der Taste <F1> erfolgt, z.B. A(-2,3.5) oder B(4.72,-8.57). Die Koordinaten werden dabei durch Beistriche getrennt. Zwischen Klein- und Großschreibung wird nicht unterschieden. Ein "=" am Anfang vor einer Punktvariablen wird ignoriert.

Viele PROGMATH-Befehle liefern als Ergebnis Zahlen oder Punkte, welche dann in den nachfolgenden Zeilen automatisch in den Text eingefügt werden. Diese Ergebniszeilen enthalten immer das Zeichen "=" und müssen am Skriptanfang mit dem Befehl "COMPRESS" aus dem Text entfernt werden. Soll eine Zeile, welche das Zeichen "=" enthält nicht gelöscht werden, dann muss am Zeilenanfang ein Punkt "." geschrieben werden.

Während eines Skriptdurchlaufs sind für die Ergebnisse bestimmte Variable reserviert. Diese sollten vorher nicht verwendet werden, weil ihre Werte dann überschrieben werden. Wird mit ihnen weitergerechnet, ist eine Umspeicherung sinnvoll. Beispielsweise durch eine Zahlenzuweisung .a = k, wo der Wert der Variablen k auf die Variable a gespeichert wird; oder durch eine Punktezuweisung UMS(A,S), wo der Punkt S mit dem neuen Namen A gezeichnet und intern abgespeichert wird.

Sieben reservierte Zahlenvariable d, k, p, q, x, y, z:
d,k ... Abschnitt und Anstieg von berechneten Geraden
p,q ... Zusatzergebnis bei speziellen Befehlen (z.B. VGG)
x,y,z ... Koordinaten von Punkten
z ... Ergebnis bei Längen, Winkeln und Flächen

Acht reservierte Punktvariable N, E, W, P, Q, S, T, Z:
N,E,W ... Nullstellen, Extremstellen, Wendestellen
P,Q ... Ergebnispunkte bei speziellen Befehlen (z.B. VGG)
S ... Schnittpunkt bei allen Schnittaufgaben
T ... Berührpunkt von Tangenten
Z ... Ergebnis bei Punkten, Vektoren oder komplexen Zahlen

(2) Die PROGMATH-Skripts

Ein PROGMATH-Skript besteht aus einer Folge von Textzeilen in denen die Mathematik-Befehle stehen. In der ersten Zeile muss immer "begin" stehen und in der letzten Zeile immer "end.". Wenn man den Mauscursor in die erste Zeile stellt und die Taste <F11> betätigt oder den Menü-Eintrag <Skript ausführen> anklickt, dann werden die nachfolgenden Befehle so lange durchlaufen bis der Befehl "end." auftritt. Auf diese Weise sind im Texteditor ein Parser und ein Interpreter integriert, welche das Erkennen und das Ausführen der Befehle bewerkstelligen.

BEISPIEL: Eine Gerade g geht durch die Punkte A(-2,1) und B(7,5). Ermittle das Lot vom Punkt C(2,8) auf die Gerade g, den Lotfußpunkt F und den Punktabstand e von der Geraden. (Die eingerückten Zeilen sind die automatisch generierten Ergebniszeilen, welche von "COMPRESS" entfernt werden).

```
begin                       // Skript-Anfang
COMPRESS                    // Zeilenkomprimierung
KOR(10,1)                   // Festlegung des Koordinatensystems
A(-2,1)                     // Speicherung und Zeichnung von Punkt A
B(7,5)                      // Speicherung und Zeichnung von Punkt B
C(2,8)                      // Speicherung und Zeichnung von Punkt C
GER(A,B)                    // Gerade durch Punkte A und B ermitteln
  y = 0.44 * x + 1.89       // Erste Ausgabezeile  (Gleichung)
  k = 0.44                  // Zweite Ausgabezeile (Anstieg)
  d = 1.89                  // Dritte Ausgabezeile (Abschnitt)
LOT(C,A,B)                  // Das Lot von C auf die Gerade g(A,B)
  y = -2.25 * x + 12.50     // Erste Ausgabezeile  (Gleichung)
  k = -2.25                 // Zweite Ausgabezeile (Anstieg)
  d = 12.50                 // Dritte Ausgabezeile (Abschnitt)
  = S(3.94,3.64)            // Vierte Ausgabezeile (Schnittpunkt)
  z = 4.77                  // Fünfte Ausgabezeile (Abstand)
.e = z                      // Umspeicherung von z auf e
UMS(F,S)                    // Umspeicherung von S auf F
  = F(3.94,3.64)            // Erste Ausgabezeile  (Punkt F)
end.                        // Skript-Ende
```

(3) Liste aller PROGMATH-Befehle

COMPRESS komprimiert alle Befehlszeilen und löscht dabei alle Zeilen, welche eine Wertzuweisung "=" enthalten. Soll eine Zeile, die das Zeichen "=" enthält, nicht gelöscht werden, dann muss am Anfang der Zeile unbedingt ein Punkt "." geschrieben werden. Dadurch werden auch Wertzuweisungen zu Variablen nachhaltig fixiert.

Der Befehl sollte immer am Skriptanfang NACH "begin" stehen. Dadurch werden die Ergebnisse eines vorangegangenen Skriptdurchlaufs gelöscht. Der Befehl funktioniert nicht im Einzelschritt-Modus mit <F1>. Im Einzelschritt-Modus muss zum Komprimieren der Skript-Zeilen der Menüpunkt <Bearbeiten-Komprimieren> gewählt werden.

CLR(n)	Grafik (n=0) und/oder Variable (n=0,n=1) löschen.
DEZ(d)	Anzahl der angezeigten Dezimalstellen d (maximal 14).
ROU(a,d)	Rundet die Zahl a intern auf d Dezimalstellen.
CUT(a,d)	Beschneidet die Zahl a intern auf d Dezimalstellen.
ZZF(a,n)	liefert von der Zahl a die Ziffer an der Stelle n.
HZF(a)	liefert den höchsten Stellenwert (10^n) der Zahl a.
MOD(a,b)	Ganzzahliger Rest bei Division a/b.
KOM(n,k)	Kombinationen (k aus n).
GGT(a,b)	größter gemeinsamer Teiler der Zahlen a,b.
KGV(a,b)	kleinstes gemeinsames Vielfache von a,b.
PFT(a)	prüft ob a eine Primzahl ist und liefert 0 oder 1.
PFZ(a)	führt eine Primfaktorenzerlegung von a durch.
ZUF(a,b)	erzeugt ganze Zufallszahlen z mit a <= z <= b. ZUF(0,1) erzeugt reelle Zahlen zwischen 0 und 1. ZUF(-1,1) erzeugt zufällig -1 oder +1.
IND(0)	liefert den Listenindex bei "zufun0" (auf Variable z)
MIN(a,b,c,...)	bestimmt das Minimum der Werte a,b,c,...
MAX(a,b,c,...)	bestimmt das Maximum der Werte a,b,c,...
KAL(Term)	berechnet den Zahlenwert des mathematischen Terms.
KOR(b,r)	löscht die Grafik und setzt die Halbbreite b des Koordinatensystems. Bei r = 1 werden die beiden Koordinatenachsen gezeichnet, bei r = 0 hingegen nicht. Bei r = 2 wird ein vollständiger Raster eingezeichnet.
RAS(b,r)	wie KOR(b,r), aber ohne Löschung der Grafik.
PEN(d,f)	setzt die Stiftdicke d (1=dünn,2=mittel,3-100=dick) und Zeichenfarbe f (-1=keine,0=weiß,1=schwarz,2=rot, 3=grün,4=blau,5=gelb,6=magenta,7=cyan,8=weiß).
PFA(f)	setzt Punkt-/Textfarbe f (-1,0,1,2,3,4,5,6,7,8).
FIL(x,y,f,r,g,b)	füllt einen mit Randfarbe f (0,1,2,3,4,5,6,7,8) begrenzten Bereich, ausgehend von dem Punkt (x,y), mit der Füllfarbe von Rot r, Grün g und Blau b (0 <= r,g,b <= 255).
PNT(P)	Anzeigen des Punktes P.
UMS(A,S)	speichert den Punkt S auf den Punkt A um, d.h. er erhält einen neuen Namen.
KXY(P)	liefert die Koordinaten x und y des Punktes P, bzw. x, y und z im dreidimensionalen Fall.
TXN(Schrift)	Angabe des Schriftnamens für die Grafik, z.B. "Arial", "Courier_New", "Symbol", "MS_LineDraw".
TXR(p)	Text ohne (p=0) oder mit (p=1) weißem Rechteck.
TXG(p)	relative Schriftgröße (von -5 bis +5, 0 = Standard).
TXF(p)	setzt Fettschrift aus (p = 0) oder ein (p = 1).
TXT(x,y,Text)	gibt an den Koordinaten (x,y) den Text aus. Blanks werden mit Underlines markiert. Ein " vor einem Buchstaben bewirkt Großschreibung.
TEX(x,y,Text)	gibt den Wert von Variablen <a> oder Punkten [P] aus.
TXZ(n,Text)	gibt in der n-ten Zeile den Text original aus (d.h. Leerstellen, Großbuchstaben und Beistriche). Mit maximal 25 Zeilen und 65 Zeichen pro Zeile.
TXV(n,Text)	gibt in der n-ten Zeile den Text original aus (d.h. Leerstellen, Großbuchstaben und Beistriche). Mit maximal 25 Zeilen und 65 Zeichen pro Zeile. Zusätzlich können im Text auch Variablenwerte und Punktkoordinaten ausgegeben werden. (Siehe dazu die Steuerungsanweisung "pause" auf Seite [09]).
LIB(A,B,Text)	beschriftet die Strecke AB mit einem Text.
WIB(A,B,C,r,Text)	beschriftet den Winkel w(ABC) mit einem Text im Abstand r vom Winkelscheitel B.

LNG(A,B)	ermittelt die Länge der Strecke AB.
WIN(A,B,C)	ermittelt den Winkel w(ABC) mit Scheitel B.
FLA(A,B,C)	ermittelt die Fläche des Dreiecks ABC.
HAP(A,B)	ermittelt den Halbierungspunkt von AB.
SWP(A,B,C)	ermittelt den Schwerpunkt von Dreieck ABC.
REC(A,C)	zeichnet ein Rechteck mit Diagonale AC.
GER(A,B)	zeichnet die Gerade y = k*x + d durch A und B. Bei Senkrechten werden x, k=10^10, d=10^10 ausgegeben!
GWG(A,B,w)	ermittelt jene Gerade durch Punkt A, die mit der Strecke AB den Winkel w bildet.
STW(A,B)	Steigungswinkel der Strecke AB.
SWI(k)	Steigungswinkel zum Anstieg k.
STP(A,B,r)	trägt vom Punkt A auf der Geraden g(A,B). in Richtung von B die Streckenlänge r ab.
LOT(A,B,C)	zeichnet die normale Gerade von Punkt A auf eine Gerade durch die Punkte B und C, und ermittelt den Lotfußpunkt F und auch den Abstand des Punktes A von der Geraden.
SSM(A,B)	zeichnet die Symmetrale der Strecke AB.
SYM(A,B,C)	zeichnet die Symmetrale des Winkels mit Scheitel B und Schenkeln BA und BC.
LIN(A,B,C...)	zeichnet einen Streckenzug durch A,B,C...
SCH(P,a,b)	Schiebung des Punktes P um den Vektor (a,b).
DRE(P,Z,w)	Drehung von P mit Zentrum Z und Winkel w.
SPI(P,A,B)	Spiegelung von P an der Achse durch A und B.
STR(P,Z,k)	Streckung von P mit Zentrum Z und Faktor k.
NSC(a,b)	Verschiebt alle Punkte eines Streckenzuges (N-Eck), der mit LIN(A,B,C,...) gezeichnet wurde, um den Vektor (a,b).
NDR(Z,w)	Dreht alle Punkte eines Streckenzuges (N-Eck), der mit LIN(A,B,C,...) gezeichnet wurde, um das Zentrum Z mit Winkel w.
NSP(U,V)	Spiegelt alle Punkte eines Streckenzuges (N-Eck), der mit LIN(A,B,C,...) gezeichnet wurde, an der Achse durch U und V.
NST(Z,k)	Streckt alle Punkte eines Streckenzuges (N-Eck), der mit LIN(A,B,C,...) gezeichnet wurde, mit dem Zentrum Z und mit dem Streckungsfaktor k.
KRS(M,r)	Kreis mit dem Mittelpunkt M und Radius r.
KRU(A,B,C)	Umkreis des Dreiecks durch die Punkte A,B,C.
KRI(A,B,C)	Inkreis des Dreiecks durch die Punkte A,B,C.
KSE(M,r,v,w)	Kreissektor mit Mittelpunkt M, Radius r, Anfangswinkel v und Endwinkel w.
KBO(M,r,v,w)	Kreisbogen (wie Kreissektor).
EBO(M,a,b,v,w)	Ellipsenbogen mit Mittelpunkt M, Halbachsen a,b und Anfangswinkel v und Endwinkel w.
BOG(M,A,B)	zieht einen Kreisbogen mit Mittelpunkt M durch die beiden Kreispunkte A und B.
RVE(n,r)	zeichnet ein regelmäßiges n-Eck mit n <= 256 und dem Umkreis mit M(0,0) und Radius r.
ELL(a,b)	Ellipse mit den Halbachsen a und b.
HYP(a,b)	Hyperbel mit den Halbachsen a und b.
PAR(p)	Parabel mit dem Parameter p.
EL2(A,B)	Ellipse durch die Punkte A und B.
HY2(A,B)	Hyperbel durch die Punkte A und B.
PA1(A)	Parabel durch den Punkt A.
PFL(A,B,d,f)	Pfeil von A nach B mit Dicke d und Farbe f. (Ein neuerliches Zeichnen löscht den Pfeil).

FUN(F(x))	zeichnet eine beliebige Funktion F(x).
FUN(F(x),a,b)	zeichnet die Funktion F(x) im Intervall [a,b].
FNA(F(x))	zeichnet die erste Ableitung einer Funktion.
FNT(F(x),n)	zeichnet n gleitende Tangenten entlang der Funktionskurve F(x) mit n = 10,11,...1000.
AFF(k,F(x))	Achsenaffinität der Funktion F(x) mit Faktor k.
BEW(u,v,w,F(x))	führt mit Funktion F(x) eine Bewegung aus, zuerst eine Drehung mit dem Winkel w um den Ursprung und dann eine Schiebung um (u/v).
GAU(z)	liefert Werte der Verteilungsfunktion F[0;z] der Standard-Normalverteilung (auf Variable f).
NUL(F(x),n,a,b)	ermittelt eine reelle Nullstelle der n-ten Ableitung der Funktion F(x) auf dem Intervall [a,b] mit n = 0,1 oder 2. Wird dort keine Nullstelle gefunden, so kann mit <Esc> abgebrochen werden.
DIF(F(x),n,a)	n-te Ableitung der Funktion F(x) an der Stelle x = a mit n = 0,1 oder 2.
INT(F(x),a,b)	bestimmtes Integral der Funktion F(x) auf dem Intervall [a,b] mit a < b. Die Funktion muss dort stetig sein und es darf kein Vorzeichenwechsel auftreten.
TKP(F(x),a)	ermittelt die Tangente an die Kurve F(x) durch den Kurvenpunkt P(a,F(a)).
TNG(F(x),a,b,c,d)	ermittelt die Tangente an die Kurve F(x) durch den Nicht-Kurvenpunkt P(a,b), wo der Berührpunkt im Intervall [c,d] liegen muss. Wird dort kein Berührpunkt gefunden, so kann mit <Esc> abgebrochen werden.
SGG(k1,d1,k2,d2)	ermittelt den Schnittpunkt der beiden Geraden y = k1*x+d1 und y = k2*x+d2.
SKG(M,r,k,d)	Schnittpunkte von Gerade y = k*x+d und Kreis mit Mittelpunkt M und Radius r.
SKK(M1,r1,M2,r2)	Schnittpunkte zweier Kreise mit den Mittelpunkten M1, M2 und den Radien r1, r2.
SFF(F1(x),F2(x),a,b)	ermittelt den Schnittpunkt der beiden Kurven F1(x) und F2(x) auf Intervall [a,b]. Wird dort kein Schnittpunkt gefunden, so kann mit <Esc> abgebrochen werden.
DET(A,B,C)	Determinante mit den Zeilenvektoren A,B,C; auch zweidimensional mit DET(A,B) möglich.
LGS(A,B,C,D)	Lineares Gleichungssystem mit den linksseitigen Zeilenvektoren A,B,C und dem rechtsseitigen Spaltenvektor D. Auch mit LGS(A,B,C) möglich.
LG4(0)	Lineares Gleichungssystem in 4 Variablen (w,x,y,z) mit den 20 Koeffizienten (a,b,..,s,t). Analog dazu sind auch LG3(0) und LG2(0) möglich.
QUA(a,b,c)	Lösungen der Gleichung $ax^2 + bx + c = 0$.
DZN(a,n)	Umwandlung der Zahl a vom 10- ins n-System.
NZD(a,n)	Umgekehrte Umwandlung mit 2 <= n <= 36.
SRD(v,w)	definiert für die Schrägrißdarstellung den Verzerrungsfaktor (v) und den Winkel (w). PFA(-1) unterbindet die Darstellung der Achsen.
SRK(P)	stellt einen dreidimensionalen Punkt P im Schrägriß als zweidimensionalen Punkt dar.

Mit Punkten als KOMPLEXE ZAHLEN kann auch gerechnet werden:

KPX(Term(A,B,C,.....))	ermittelt den Wert des Terms mit den komplexen Zahlen A,B,C,..... Zulässige Operatoren: +,-,*,/,^,(,). Nach dem Potenzoperator ^ muss ein reeller Rechenausdruck folgen, z.B. KPX(A+B^(1/2)).
ADD(A,B)	Addition zweier komplexer Zahlen.
SUB(A,B)	Subtraktion zweier komplexer Zahlen.
MUL(A,B)	Multiplikation zweier komplexer Zahlen.
DIV(A,B)	Division zweier komplexer Zahlen.
POT(A,n)	n-te Potenz der komplexen Zahl A.
WUR(A,n)	n-te Wurzel der komplexen Zahl A.
POL(A)	liefert die Polarkoordinaten (r,w) von A.
BIN(r,w)	Kartesische Koordinaten (x,y) von (r,w).

Mit Punkten als VEKTOREN kann ebenfalls gerechnet werden.
Dabei werden zwei oder drei Vektor-Koordinaten eingegeben.
Die Anzahl der Koordinaten wird automatisch erkannt.

VEK(Term(A,B,C,.....))	ermittelt den Wert des Terms mit den Vektoren A,B,C,..... Dabei zulässige Operatoren: +,-,*,(,). Der Operator * bedeutet die Multiplikation eines Vektors mit einem reellen Ausdruck, z.B. VEK(A+B*(1/2)).
VSU(A,B)	Vektorsumme A+B.
VDI(A,B)	Vektordifferenz A-B.
VFA(A,k)	Vektorvielfaches k*A.
VPR(A,B)	Vektorprodukt von A und B.
VSK(A,B)	Skalarprodukt von A und B.
VLE(A)	Länge eines Vektors A.
VWI(A,B)	Winkel zwischen A und B.
VFL(A,B)	Fläche zwischen A und B (Parallelogramm).
VOL(A,B,C)	Volumen zwischen A, B, C (Parallelepiped).
VEV(A)	Normierter Vektor von A (Einheitsvektor).
VNV(A)	Normale Vektoren auf A.
VNA(P,A,N)	Abstand des Punktes P von der Geraden (2-dim) oder Ebene (3-dim) durch Punkt A mit dem Normalvektor N. Außerdem wird der Lotfußpunkt F ermittelt.
VGG(A,B)	Geradengleichungen im Raum durch die Punkte A,B. Ausgabe: <funtex1> und <funtex2> und die beiden Normalvektoren P und Q und die Konstanten p und q.
VEG(A,B,C)	Ebenengleichung im Raum durch die Punkte A,B,C. Ausgabe: <funtex1> und Normalvektor Z und Konstante z.
VNG(A,N)	Gleichung der Geraden (2-dim) oder Ebene (3-dim) durch den Punkt A mit dem Normalvektor N. Ausgabe: <funtex1> und Normalvektor Z und Konstante z.
SCS(0)	Zwischenspeicherung der aktuellen Grafik.
SCR(0)	Wiederherstellung der zwischengespeicherten Grafik.
LPC(Name,g)	Lädt eine JPEG-Datei "Name" in das Koordinatenfenster, mit originaler Größe (g=0) oder angepasst (g=1).
PAG(0)	Markiert im einen Seiten-Anfang. Dann kann man beim "pause"-Befehl mit <F2><Enter> eine Seite zurückgehen.
SXY(n)	Speichert die Variablen x und y auf die internen Listenelemente DatX[n] und DatY[n]. Die Elemente DatX[0] und DatY[0] enthalten die Datenanzahl (<=100).
WXY(0)	Schreibt die interne Liste in ein CSV-Textfile.
RXY(0)	Liest ein CSV-Texfile in die interne Liste
LXY(n)	Lädt die internen Listenelemente DatX[n] und DatY[n] auf die Variablen x und y.
DXY(0)	Löscht alle internen Listenelemente.

(4) STEUERUNGS-Anweisungen

Neben diesen 140 Mathematikbefehlen gibt es noch 36 Anweisungen zur Steuerung des Skriptablaufes. Eine Steuerungsanweisung ist bereits oben genau beschrieben worden, nämlich "compress".

begin	Erste Zeile eines PROGMATH-Skripts
end.	Letzte Zeile eines PROGMATH-Skripts
wait(t)	Wartet t Millisekunden im Skriptablauf.
pause(Text)	Erzeugt eine Pause und zeigt den Text an.

Wenn im Pausentext eine Variable zwischen spitzen Klammern < > steht, dann wird dort ihr Zahlenwert angezeigt. Wenn im Text ein Punktname zwischen eckigen Klammern [ ] steht, dann werden dort die Punktkoordinaten angezeigt, dreidimensional mit { }. Das Zeichen & im Pausentext bewirkt einen Zeilenumbruch.

Mit "pause(<funtex1>)" wird der Wert der internen Textvariablen "funtex1" angezeigt. Hingegen wird mit "pause(<funtex2>)" der Wert der zweiten Textvariablen "funtex2" angezeigt.

Beim Ausgabebefehl "pause" und bei den Eingabebefehlen "invar", "input2", "input3", "intex1" und "intex2" kann mit Taste <Esc> das Skript jederzeit abgebrochen werden. Beim Befehl "pause" kann die aktuelle Grafik gedruckt werden (mit Funktionstaste <F4>).

select(Text: Var,N)	Erzeugt eine Auswahlbox mit N Schaltern von 1 bis N. Der Auswahlschalter 0 ist immer vorhanden. Die ausgewählte Nummer wird auf die Variable Var gespeichert. Zusätzlich wird der Text angezeigt.
invar(Text: Variablenliste)	Eingabe von Variablenwerten.
	Wenn dieser Text ein Fragezeichen "?" enthält, dann wird mit einem Doppelklick in das Eingabefeld ein Taschenrechner aufgerufen. Wird er geschlossen, dann wird das Rechenergebnis in das Eingabefeld übertragen. Diese Technik wird bei interaktiven Fragen/Antworten verwendet. Der Befehl "calcoff" sperrt den Rechner. (siehe dazu die Beispiele ab Seite -26-).
input2(Punktname)	Eingabe der Koordinaten x,y einer Punkt- bzw. Vektor-Variablen (zweidimensional).
input3(Punktname)	Eingabe der Koordinaten x,y,z einer Punkt- bzw. Vektor-Variablen (dreidimensional).
ifle(Var,W,Ziel)	Vergleicht die Variable Var mit dem Wert W. Dabei kann W auch eine Variable sein. Wenn Var <= W ist, erfolgt ein Sprung zu jener Befehlszeile, wo der Ziel-Text steht. Dort muss diesem Ziel-Text ein Underline "_" vorangestellt werden (symbolische Adresse). Wenn aber Var > W ist, wird weiter gegangen.
ifls(Var,W,Ziel)	Wie "ifle", aber nur wenn Var < W ist.
ifge(Var,W,Ziel)	Wie "ifle", aber nur wenn Var >= W ist.
ifgr(Var,W,Ziel)	Wie "ifle", aber nur wenn Var > W ist.
ifeq(Var,W,Ziel)	Wie "ifle", aber nur wenn Var = W ist.
ifne(Var,W,Ziel)	Wie "ifle", aber nur wenn Var <> W ist.

iferror(Ziel)	Wie "ifeq", aber nur bei internem Fehler.
delerror	Setzt die interne Fehlervariable auf 0.
seterror	Setzt die interne Fehlervariable auf 1.

Wenn bei einem mathematischen Befehl ein Fehler auftritt, dann wird die interne Fehlervariable, welche normalerweise den Wert Null hat, automatisch auf 1 gesetzt. Sie kann dann mit "iferror" abgefragt und mit "delerror" wieder auf Null gesetzt werden.

goto(Ziel)	Springt unbedingt zu jener Befehlszeile, die durch das "Ziel" symbolisch adressiert ist. Im Listing muss vor den Namen des Sprungziels ein Underline gestellt werden ("_Ziel").
exit	Bricht das Programm unbedingt ab.

Neben den Steuerungsanweisungen für den Verlauf des PROGMATH-Skripts geibt es noch so genannte Stringbefehle. Sie beziehen sich auf die Zuweisung, Eingabe, Speicherung und Verarbeitung von zwei internen Stringvariablen "funtex1" und "funtex2". Diese können mit gültigen mathematischen Ausdrücken oder Funktionstermen belegt werden und sie versorgen dann folgende Befehle als Eingabeparameter:

NZD,KAL,KPX,VEK,TXT,TXZ,TXV,DIF,INT,FUN,FNA,BEW,NUL,TKP,TNG,SFF.

Zur Datenausgabe werden sie bei folgenden Befehlen verwendet: DZN,PFZ,QUA,VNG,VGG,VEG.

setfun1(Formel)	Weist der Textvariablen "funtex1" eine Formel zu.
setfun2(Formel)	Weist der Textvariablen "funtex2" eine Formel zu. Hier können auch die 4 Namen "funtex1", "-funtex1", "funtex2" und "-funtex2" eingegeben werden.
intex1(Text)	Eingabe einer mathematischen Formel und Speicherung auf der Textvariablen "funtex1". Dabei wird der Text angezeigt.
intex2(Text)	Eingabe einer mathematischen Formel und Speicherung auf der Textvariablen "funtex2". Dabei wird der Text angezeigt.
stofun1	Speichert intern zusätzlich den Text von "funtex1".
getfun1	Holt den mit "stofun1" gespeicherten Text zurück.
stofun2	Speichert intern zusätzlich den Text von "funtex2".
getfun2	Holt den mit "stofun2" gespeicherten Text zurück.
fun1q	Bildet f(x)^2 mit der Formel f(x) in "funtex1".
fun1x	Bildet x*f(x) mit der Formel f(x) in "funtex1".
fun1m	Bildet x*f(x)^2 mit der Formel f(x) in "funtex1".
	Die Ergebnisse sind auf der Variablen z gespeichert.
IBO(l,r)	Bildet das Integral von sqrt(1+f'(x)^2) mit f(x) in "funtex1" zwischen den Grenzen l und r. Das wird zur Berechnung von Bogenlängen verwendet.
IBX(l,r)	Bildet das Integral von x*sqrt(1+f'(x)^2) mit f(x) in "funtex1" zwischen den Grenzen l und r. Das wird zur Berechnung von Bogenschwerpunkten verwendet.
IBY(l,r)	Bildet das Integral von f(x)*sqrt(1+f'(x)^2) mit f(x) in "funtex1" zwischen den Grenzen l und r. Das wird zur Berechnung der Mantelfläche von Drehkörpern verwendet.
	Die Ergebnisse sind auf der Variablen z gespeichert.

ifrep(Ziel)	Wenn bei "intex1" oder "intex2" der Buchstabe "w" eingegeben wird, dann wird automatisch der zuletzt eingegebene Text aktualisiert (Wiederholungstaste). Dann wird mit dem Befehl zu einem Ziel gesprungen.
delrep	Löscht diese Wiederholungsfunktion zur Neu-Eingabe.
zufun1	Wählt aus einer Liste von mit Kommas getrennten Texten (Formeln), welche mittels "setfun1(Liste)" angelegt wird, zufällig einen Text (Formel) aus und speichert ihn auf die Variable "funtex1".
zufun2	Wählt aus zwei Listen, die mit "setfun1(Liste)" und "setfun2(Liste)" erzeugt sind, eine zufällige Position aus und speichert die synchronen Werte auf "funtex1" und "funtex2".
zufun0	Wählt aus zwei Listen "setfun1(Liste1)" und "setfun2(Liste2)" schrittweise zwei Texte (Formeln) aus, die in beiden Listen an der gleichen Position stehen. Diese synchronen Texte werden dann auf "funtex1" und "funtex2" gespeichert. Die Auswahl beginnt mit dem ersten Listeneintrag und durchläuft die ganze Textliste. Am Ende beginnt der Durchlauf wieder am Listenanfang.
ind(0)	Mit dem Befehl "ind(0)" wird die aktuelle Listenposition der Variablen "z" zugewiesen.

```
begin
// Demo-Skript zum Befehl "zufun0"
_anf
clr(0)
dez(4)
txr(1)
_start
kor(10,1)
compress
setfun1(2*x+3,0.5*x^2,1/x,sqrt(x),sqrt(25-x^2),3*sin(deg(x)))
setfun2(2,x,-1/x^2,0.5/sqrt(x),-x/sqrt(25-x^2),3*cos(deg(x)))
zufun0
ind(0)
.n = z
pfa(1)
txg(2)
txv(1,<n>-te Funktion)
txg(0)
txv(24,Funktion: f(x) = <funtex1>          )
pen(2,1)
fun(funtex1)
pause(Weiter zur Ableitung)
pfa(2)
txv(25,Ableitung: f'(x) = <funtex2>        )
pen(2,2)
fun(funtex2)
pause(<n>. Funktion (von 6))
ifeq(n,6,aus)
goto(start)
_aus
pause(Wiederholen oder Abbruch mit "Esc")
goto(anf)
end.
```

Mit diesen Steuerungsanweisungen können Wertevergleiche, unbedingte und bedingte Sprünge, Wiederholungsschleifen, Dateneingaben und Datenausgaben und verschiedene Zusatzfunktionen realisiert werden.

---

(5) Zehn Skript-Beispiele

[S1] Ein einfaches geometrisches Skript-Beispiel

Das folgende erste Listing zeigt den komprimierten Programmcode vor der Ausführung des Skripts.

```
begin
// Darstellung des Inkreises eines Dreiecks
compress
kor(10,2)
A(-2,-2)
B(8,3)
C(2,8)
lin(A,B,C,A)
kri(A,B,C)
lot(I,A,B)
pen(3,2)
lin(I,S)
end.
```

Das folgende zweite Listing zeigt den Programmcode nach der Ausführung des Skripts. Alle Rechenergebnisse der einzelnen Befehle sind dabei sichtbar nach rechts eingerückt, und sie können im Skriptablauf weiter verwendet werden. Bei jeder Skriptausführung werden sie am Skriptanfang durch den Befehl "compress" gelöscht.

```
begin
// Darstellung des Inkreises eines Dreiecks
compress
kor(10,2)
A(-2,-2)
B(8,3)
C(2,8)
lin(A,B,C,A)
kri(A,B,C)
   = I(3.12,3.57)
   r = 2.69
lot(I,A,B)
   y = -2.00 * x + 9.81 = -10.27
   k = -2.00
   d = 9.81
   = S(4.32,1.16)
   z = 2.69
pen(3,2)
lin(I,S)
end.
```

.

[S2] Ein einfaches arithmetisches Skript-Beispiel

```
begin
// Ein Programm mit
// zwei unbedingten Sprüngen "goto(start)" und "goto(weiter)"
// und mit einem bedingten Sprung "ifeq(b,0,fehler)".
// Jedes Sprungziel steht allein in einer Textzeile
// und beginnt mit einem Underline (z.B. "_start").

 _start
compress
clr(0)
dez(2)
kor(10,0)
txv(1,Einfache Rechnungen)
.a = 3
.b = 0
invar(Zwei Zahlen a und b eingeben: a,b)
txv(2,a = <a>, b = <b>)
.c = a + b
.d = a - b
.e = a * b
txv(4,<a> + <b> = <c>)
txv(5,<a> - <b> = <d>)
txv(6,<a> * <b> = <e>)
ifeq(b,0,fehler)
.f = a / b
txv(7,<a> / <b> = <f>)
goto(weiter)

_fehler
txv(7,<a> / <b> = ? Division durch Null ist nicht möglich !)

_weiter
pause(Wiederholen oder Beenden mit "Esc")
goto(start)

end.
```

Hinweis: Alle Skripts in diesem Hilfetext können markiert und mit mit <Strg C> und <Strg V> in den Editor kopiert werden. Dann werden mit <Bearbeiten-Linker Rand> die n leeren linken Spalten gelöscht (z.B. mit n = -8). Zusätzlich müssen auch etwaige Zeilen, die Seitenangaben enthalten, mit <Strg Y> händisch gelöscht werden. Zuletzt kann das Skriptum ausgeführt werden.

```
================================================================
[S3] Das Skript ermittelt den Umkreis eines Dreiecks
     (mit eingerückten Rechenergebnissen)
================================================================

begin
_start
compress
clr(0)
dez(2)
.g = 10
kor(g,1)
txv(1,Dreieck und Umkreis)
invar(Halblänge des Koordinatensystems: g)
kor(g,1)
txv(1,Dreieck und Umkreis)
pfa(-1)
A(1,3)
B(4,0)
C(8,8)
pfa(1)
input2(A)
input2(B)
input2(C)
lin(A,B,C,A)
lng(A,B)
     z = 4.24
.c = z = 4.24
lng(A,C)
     z = 8.60
.b = z = 8.60
lng(B,C)
     z = 8.94
.a = z = 8.94
.u = a + b + c = 21.78
.s = u/2 = 10.89
.d = s*(s-a)*(s-b)*(s-c) = 323.38
ifgr(d,0,weiter)
pause(A,B,C bilden kein Dreieck!)
goto(start)

_weiter
.f = sqrt(d)
win(B,A,C)
     z = 80.54
.i = z = 80.54
win(A,B,C)
     z = 71.57
.j = z = 71.57
win(A,C,B)
     z = 27.90
.k = z = 27.90
.s = i + j + k = 180.01
rou(s,0)
kru(A,B,C)
     = U(5.33,4.33)
     r = 4.53
pen(2,2)
lin(A,U)
lib(A,U,r)
pen(1,1)
```

```
pfa(4)
txv(19,Dreieck ABC mit [A], [B], [C])
pfa(1)
txv(20,c = AB = <c>, b = AC = <b>, a = BC = <a>)
txv(21,Umfang = a + b + c = <u> cm)
txv(22,Fläche F = <f> cm²)
txv(23,w(B,A,C) = <i>°, w(A,B,C) = <j>°, w(A,C,B) = <k>°)
txv(24,Winkelsumme = <s>°)
pfa(2)
txv(25,Umkreis: [U], r = <r>)
pfa(1)
pause(Wiederholen oder Beenden mit "Esc")
goto(start)
end.

=====================================================================
[S4] Ein Skript-Beispiel mit einem selektiven Auswahlmenü
     für die drei Grundaufgaben der Prozentrechnung
=====================================================================

begin
clr(0)

_start
compress
kor(g,0)
txg(4)
pfa(4)
txz(2,Rechnen mit Prozenten)
pfa(1)
txg(0)
txz(4,(alle Zahlen auf 2 Dezimalstellen runden))
txz(7,1 Prozent (%) entspricht 1 Hundertstel vom Grundwert G.)
txz(8,Der Teil vom Grundwert G, der  p  Prozenten entspricht, heißt)
txz(9,Anteil A. Dafür gilt die Schlussrechnung  A = (G / 100) * p.)
select([1] Anteil  [2] Prozente  [3] Grundwert  [0] Ende: w,3)
ifeq(w,1,teil)
ifeq(w,2,proz)
ifeq(w,3,grund)
ifeq(w,0,aus)
goto(start)

_teil
zuf(50,200)
.g = z
zuf(1,100)
.p = z
.a = (g / 100) * p
txz(12,Gegeben sind Grundwert und Prozente.)
pfa(4)
txv(14,Grundwert  G = <g> €)
txv(15,Prozente  p = <p> %)
txz(16,Anteil  A = ?)
pfa(1)
txz(19,Gesucht ist jener Anteil A, welcher den)
txz(20,p Prozenten vom Grundwert G entspricht.)
pause(Weiter zum Ergebnis)
pfa(2)
txv(22,Anteil  A = (G / 100) * p  =  <a> €)
pfa(1)
pause(Zurück)
goto(start)
```

```
_proz
zuf(50,200)
.g = z
zuf(10,g)
.a = z
.p = (100 / g) * a
txz(12,Gegeben sind Grundwert und Anteil.)
pfa(4)
txv(14,Grundwert  G = <g> €)
txv(15,Anteil  A = <a> €)
txz(16,Prozente  p = ?)
pfa(1)
txz(19,Gesucht sind jene Prozente p, denen)
txz(20,der Anteil A vom Grundwert G entspricht.)
pause(Weiter zum Ergebnis)
pfa(2)
txv(22,Prozente  p = (100 / G) * A  =  <p> %)
pfa(1)
pause(Zurück)
goto(start)

_grund
zuf(50,200)
.a = z
zuf(1,100)
.p = z
.g = (a / p) * 100
txz(12,Gegeben sind Anteil und Prozente.)
pfa(4)
txv(14,Anteil  A = <a> €)
txv(15,Prozente  p = <p> %)
txz(16,Grundwert  G = ?)
pfa(1)
txz(19,Gesucht ist jener Grundwert G, dessen)
txz(20,Anteil A den p Prozenten entspricht.)
pause(Weiter zum Ergebnis)
pfa(2)
txv(22,Grundwert  G = (A / p) * 100  =  <g> €)
pfa(1)
pause(Zurück)
goto(start)

_aus
end.
```

```
================================================================
[S5] Das Skript ermittelt Linearkombinationen von zwei Vektoren
================================================================

begin
_start
compress
clr(0)
dez(2)
kor(10,2)
pfa(1)
txg(2)
txz(1,Linearkombinationen von Vektoren)
txg(0)
txz(3,c´ = j*a´ + k*b´)
pfa(-1)
O(0,0)
A(3,1)
B(2,4)
setfun1((-3)*A + (1/2)*B )
pfa(1)
input2(A)
lin(O,A)
input2(B)
lin(O,B)
txv(24,a´ = [A], b´ = [B])
intex1
vek(funtex1)
ums(C,Z)
kxy(A)
.a = x
.b = y
kxy(B)
.c = x
.d = y
kxy(C)
.e = x
.f = y
vle(A)
.u = z
vle(B)
.v = z
vle(C)
.w = z
max(u,v,w)
.g = round(z) + 2
kor(g,2)
txg(2)
txz(1,Linearkombinationen von Vektoren)
txg(0)
txz(3,c´ = j*a´ + k*b´)
ums(A,A)
ums(B,B)
pen(1,2)
O(0,0)
lin(O,A)
lib(O,A,a´)
lin(O,B)
lib(O,B,b´)
txv(24,a´ = [A], b´ = [B])
```

```
ums(C,C)
pen(2,2)
lin(O,C)
lib(O,C,c´)
pfa(2)
txv(25,c´ = <funtex1> = [C])
pfa(1)
pause(Wiederholen oder Beenden mit "Esc")
goto(start)
end.

=====================================================================
[S6] Das Skript ermittelt die Fläche unter einer Kurve f(x)
     und deren Flächenschwerpunkt
=====================================================================

begin
_start02
compress
clr(0)
dez(4)
kor(10,1)
txr(0)
pfa(4)
txg(2)
txz(1,Inhalt und Schwerpunkt von Flächen)
txg(0)
pfa(1)
setfun1(sqrt(25-x^2))
stofun1
.a = 0
.b = 5
.g = 10
invar(Halbbreite des Koordinatensystems: g)
_anf02
compress
kor(g,1)
pfa(4)
txz(1,Inhalt und Schwerpunkt von Flächen)
pfa(1)
txz(2,Gegeben ist eine Funktion y = f(x) auf dem Intervall [a;b].)
txz(3,Das Zeichen ∫ bedeutet das Integral der Funktion f(x).)
txz(4,Für den Schwerpunkt S(u/v) der Fläche F = F(a;b) gilt:)
pfa(2)
txg(2)
txz(06,F = ∫ f(x)dx )
txz(08,u = (1/F) * ∫ x*f(x)dx )
txz(10,v = 1/(2*F) * ∫ f(x)²dx)
txg(0)
pfa(1)
getfun1
intex1(Funktion f(x) eingeben)
stofun1
txv(14,Funktion: y = f(x) = <funtex1>)
pen(1,1)
fun(funtex1)
invar(Grenzen a und b mit eingeben: a,b)
ifle(b,a,anf02)
txv(15,Grenzen:  a = <a> und b = <b>)
.l = a
.r = b
```

```
pen(3,4)
fun(funtex1,a,b)
pen(2,1)
pfa(-1)
A(a,0)
dif(funtex1,0,l)
iferror(fehl02)
.c = z
dif(funtex1,0,r)
iferror(fehl02)
.d = z
txv(16,f(a) = <c> ,  f(b) = <d>)
C(a,c)
B(b,0)
D(b,d)
pfa(1)
lin(A,C)
lin(A,B)
lin(B,D)
pen(1,1)
txt(a,0,a)
txt(b,0,b)
pause(Weiter)
int(funtex1,l,r)
iferror(fehl02)
.f = z
getfun1
fun1q
int(funtex1,l,r)
iferror(fehl02)
.v = z / (2*f)
.i = pi * z
getfun1
fun1x
int(funtex1,l,r)
iferror(fehl02)
.u = z / f
S(u,v)
pfa(-1)
E(u,0)
F(0,v)
pfa(1)
pen(1,2)
lin(E,S)
lin(F,S)
pfa(2)
lib(E,S,v)
lib(F,S,u)
txv(18,Schwerpunkt:  [S])
txv(19,Fläche F = <f>)
pause(Weiter)
pfa(4)
txv(21,Volumen des Drehkörpers:  V = ¶ * ƒ y²dx = <i>)
txv(22,Volumen des Drehkörpers:  V = F * (2*¶*v) = <i>)
pfa(1)
pen(1,1)
txz(23,Erste Guldin-Regel:)
txz(24,Das Volumen des Drehkörpers ist gleich dem Produkt aus der)
txz(25,erzeugenden Fläche und der Kreisbahn des Schwerpunktes.)
```

```
_wied02
.w = 0
invar((1) Wiederholen   (2) Neustart   (0) Beenden: w)
ifeq(w,0,aus02)
ifeq(w,1,anf02)
ifeq(w,2,start02)
goto(wied02)
_fehl02
delerror
pause(Fehler ... )
goto(start02)
_aus02
dez(2)
end.


======================================================================
[S7] Kurvendiskussion von Funktionen y = f(x)
======================================================================

begin
clr(0)
dez(4)
compress
kor(10,0)
pfa(2)
txg(4)
txz(2,KURVENDISKUSSION)
txg(0)
txr(1)
pfa(1)
txz(4,Hinweis: Eingabe von beliebigen Funktionen  y = f(x). )
txz(5,Zulässige Operatoren in f(x):  (, ), +, -, *, /, ^, sqrt, abs,)
txz(6,round,trunc,pi, exp,ln, deg,sin,cos,tan,acos,asin,atan.)
txz(8,Wahl der Halbbreite des Koordinatensystems g,)
txz(9,so dass gilt:  -g <= x <= +g  und  -g <= y <= +g.)
txz(11,Zur Berechnung der Null-, Extrem- und Wendestellen )
txz(12,der Funktion, muss vorher immer ein Intervall für die)
txz(13,Stellen angegeben werden. Wenn dort keine der ge- )
txz(14,suchten Stellen vorhanden ist, dann wird der Such-)
txz(15,prozess mit der Taste <Esc> abgebrochen.)
txz(17,Zusätzlich können für jeden x-Wert der y-Wert  y = f(x), )
txz(18,die erste und die zweite Ableitung  y'(x)  und  y''(x)  und )
txz(19,die Kurventangente im Punkt P(x/y) berechnet werden. )
txz(21,Hinweis: Nullstellen, wo die Kurve die x-Achse berührt)
txz(22,und nicht schneidet, werden nicht immer erkannt !)
setfun1(x^3-2*x^2-3*x+4)

_eingabe
.g = 10
invar(Halbbreite des Koordinatensystems (2..100): g)
ifls(g,2,eingabe)
ifgr(g,100,eingabe)

intex1(Funktion f(x))
.w = 0
```

```
_anf
compress
kor(g,2)
pen(2,2)
fun(funtex1)
pfa(1)
txv(1,f(x) = <funtex1>)
pen(1,1)
invar(Null-,Extrem-,Wendestellen(0,1,2), y-Wert(3), Neustart(4), Ende(5): w)
ifeq(w,0,nulli)
ifeq(w,1,extri)
ifeq(w,2,wendi)
ifeq(w,3,wert)
ifeq(w,4,eingabe)
ifeq(w,5,schluss)
goto(anf)

_nulli
.l =-g
.r = g
invar(X-Intervall (l,r) für Nullstelle: l,r)
.s = w
nul(funtex1,s,l,r)
iferror(fehl)
txv(23,Nullstelle [N])
pause(Weiter)
goto(anf)

_extri
.l = -g
.r = g
invar(X-Intervall (l,r) für Extremstelle: l,r)
.s = w
nul(funtex1,s,l,r)
iferror(fehl)
txv(23,Extremstelle [E])
kxy(E)
E(x,y)
pause(Weiter)
goto(anf)

_wendi
.l = -g
.r = g
invar(X-Intervall (l,r) für Wendestelle: l,r)
.s = w
nul(funtex1,s,l,r)
iferror(fehl)
kxy(W)
.u = x
.v = y
pen(1,-1)
tkp(funtex1,u)
pen(1,1)
ums(W,T)
txv(23,Wendestelle [W] )
txv(24,Wendetangente y = (<k>)*x+(<d>) )
pause(Weiter)
goto(anf)
```

```
_wert
.l = 0
invar(x-Koordinate von f(x): l)
dif(funtex1,0,l)
iferror(fehl)
.y = z
P(l,y)
dif(funtex1,1,l)
iferror(fehl)
.u = z
dif(funtex1,2,l)
.v = z
tkp(funtex1,l)
ums(P,T)
txv(21, Punkt [P] )
txv(22, f(<l>) = <y> )
txv(23, f'(<l>) = <u> )
txv(24, f''(<l>) = <v> )
txv(25, Tangente y = <k>*x + <d> )
pause(Weiter)
goto(anf)

_fehl
pause(Unzulässiger Wert)
goto(anf)

_schluss
end.
```

```
================================================================
[S8] Das Skript erzeugt einen dreidimensionalen Quader
================================================================

begin
_start
compress
clr(0)
dez(2)
kor(10,0)
pfa(2)
txg(15)
txz(12,       Q U A D E R)
txg(0)
txr(0)
pfa(1)

_wied
.s = 1
invar((1) Mit Zufall   (2) Mit Eingaben: s)
ifeq(s,1,zufa)
ifeq(s,2,eing)
goto(wied)

_eing
.a = 6
.b = 6
.c = 6
invar(Kanten a, b und c eingeben: a,b,c)
ifle(a,0,eing)
ifle(b,0,eing)
ifle(c,0,eing)
max(a,b,c)
.g = z + 2
kor(g,1)
pfa(4)
txg(2)
txz(1,Quader)
txg(0)
txr(0)
goto(show)

_zufa
.g = 10
kor(g,0)
pfa(4)
txg(2)
txz(1,Quader)
txg(0)
pfa(1)
.g = 10
.i = g/5
.j = g-1
zuf(i,j)
.a = z
zuf(i,j)
.b = z
zuf(i,j)
.c = z
goto(show)
```

```
_show
.v = 1/2
.w = 40
srd(v,w)
pfa(-1)
pen(1,1)
A(b,0,0)
B(b,a,0)
C(0,a,0)
D(0,0,0)
E(b,0,c)
F(b,a,c)
G(0,a,c)
H(0,0,c)
srk(A)
srk(B)
srk(C)
srk(D)
srk(E)
srk(F)
srk(G)
srk(H)
pen(2,1)
lin(A,B,F,E,A)
lin(B,C,G,F,B)
lin(E,F,G,H,E)
swp(A,B,F)
kxy(Z)
fil(x,y,1,250,220,220)
swp(B,C,F)
kxy(Z)
fil(x,y,1,250,220,220)
swp(E,F,H)
kxy(Z)
fil(x,y,1,230,200,200)
pen(1,1)
lin(A,B,C,D,A)
lin(A,D,H,E,A)
lin(D,C,G,H,D)
pen(1,2)
lin(A,C)
lin(A,G)
lin(C,G)
swp(A,C,G)
kxy(Z)
fil(x,y,2,220,210,230)
pfa(1)
pen(2,1)
lin(B,F,G,C,B)
lin(A,B,F,E,A)
lib(A,G,r)
lib(A,B,a)
lib(B,C,b)
lib(C,G,c)
pen(1,1)
```

```
txv(3,Gegeben:)
pfa(4)
txv(4,Seite  a = <a> cm)
txv(5,Seite  b = <b> cm)
txv(6,Seite  c = <c> cm)
pfa(1)
txz(7,Gesucht:)
pfa(4)
txz(8,Raumdiagonale  r)
txz(9,Volumen  V)
txz(10,Oberfläche  O)
pfa(1)
.v = a*b*c
.d = sqrt(a*a + b*b)
.r = sqrt(a*a + b*b + c*c)
.o = 2*a*b + 2*a*c + 2*b*c
pause(Weiter zum Ergebnis)
pfa(2)
txv(22,Raumdiagonale:  r² = a² + b² + c²,  r = <r> cm)
txv(23,Volumen:  V = a*b*c = <v> cm³)
txv(24,Oberfläche:  O = 2*(a*b + a*c + b*c) = <o> cm²)
pfa(1)
pause(Wiederholen oder Beenden mit "Esc")
goto(start)
end.
```

[S9] Ein Skript-Beispiel MIT interaktiven Fragen und Antworten mit Verwendung des Befehls „invar(Produkt = ?: x)“

```
begin
calcoff
clr(0)
.m = 10
.n = 0
.r = 0

_start
compress
dez(0)
.n = n + 1 = 1
kor(10,0)
txr(1)
pfa(1)
txg(2)
txv(2,<n>.-ter Versuch von maximal <m> Versuchen)
txg(8)
pfa(4)
txv(10,   Ein-Mal-Eins)
zuf(1,10)
     z = 5
.a = z = 5
zuf(1,10)
     z = 8
.b = z = 8
pfa(2)
txv(13,   <a> * <b> = ?)
.c = a * b = 40
.x = 0
invar(Produkt = ?: x)
// interaktiver Frage/Antwort-Eingabebefehl
pfa(1)
txv(15,   <a> * <b> = <c>)
pfa(1)
txg(2)
ifeq(x,c,richtig)
goto(falsch)

_richtig
.r = r + 1 = 1
txv(24,<r> von <n> Antworten richtig)
pause(<x> = richtige Antwort)
ifeq(n,m,aus)
goto(start)

_falsch
txv(24,<r> von <n> Antworten richtig)
pause(<x> = falsche Antwort)
ifeq(n,m,aus)
goto(start)

_aus
pfa(2)
txg(2)
.p = 100 / n * r
rou(p,0)
txv(24,<n> Versuche, <r> richtig . . . Ergebnis: <p>%)
pause(Ende - Drucken mit "F4")

end.
```

[S10] Ein Skript-Beispiel MIT interaktiven Fragen und Antworten

```
begin
clr(0)
.n = 0
.m = 0

_start
.n = n + 1 = 1.00
compress
dez(2)
kor(10,0)
.g = 10
.j = g/2 = 5.00
.i = 3*g/2 = 15.00
zuf(j,i)
     z = 12
.a = z = 12.00
.i = 3*a/5 = 7.20
.j = a/5 = 2.40
zuf(j,i)
     z = 3
.c = z = 3.00
.j = g/5 = 2.00
.i = 3*g/4 = 7.50
zuf(j,i)
     z = 8
.h = z = 8.00
kor(g,0)
txr(0)
pfa(4)
txg(2)
txz(1,Das gleichschenkelige Trapez)
txg(0)
.s = (a - c) / 2 = 4.50
.t = a - s = 7.50
.r = a + c = 15.00
.b = sqrt(h*h + s*s) = 9.18
.e = sqrt(h*h + t*t) = 10.97
.f = r*h/2 = 60.00
pen(2,1)
.u = a/2 = 6.00
.v = c/2 = 1.50
A(-u,0)
B(u,0)
C(v,h)
D(-v,h)
E(v,0)
F(-v,0)
lin(A,B,C,D,A)
pen(1,1)
lin(E,C)
lin(A,C)
lin(F,D)
pfa(-1)
swp(E,B,C)
    = Z(3.00,2.67)
kxy(Z)
    x = 3.00
    y = 2.67
fil(x,y,1,250,200,200)
```

```
pfa(2)
txr(0)
lib(A,B,a)
lib(C,D,c)
lib(B,C,b)
lib(A,D,b)
lib(A,C,e)
lib(E,B,x)
lib(E,C,h)
pfa(1)
txv(4,Gegeben:)
pfa(4)
txv(6,Seite a = <a> cm)
txv(7,Seite c = <c> cm)
txv(8,Höhe  h = <h> cm)
pfa(1)
pfa(2)
txz(16,Gesucht: Seite b, Diagonale e, Fläche F)
pfa(1)
txz(17,Ergebnisse auf 2 Dezimalen gerundet.)
pfa(4)
txv(19,(1)  Strecke x ausrechnen)
txv(20,(2)  b und e aus den Dreiecken EBC und AEC berechnen)
txz(21,(3)  Fläche  F = h * (a + c) / 2  ausrechnen)
.p = 0
.q = 0
.w = 0
invar(Seite b, Diagonale e, Fläche F = ?: p,q,w)
// interaktiver Frage/Antwort-Eingabebefehl
pfa(2)
.x = s = 4.50
txv(23,b = <b> cm, e = <e> cm, F = <f> cm²)
pfa(1)
// Rundung der Ergebnisvariablen auf 2 Dezimalen
rou(p,2)
rou(q,2)
rou(w,2)
rou(b,2)
rou(e,2)
rou(f,2)
.i = p * q * w = 6042.28
.j = b * e * f = 6042.28
ifeq(i,j,richtig)
goto(falsch)

_richtig
.m = m + 1 = 1.00
.z = m * 100 / n = 100.00
rou(z,0)
txv(25,<n> Versuche, <m> richtig . . . Ergebnis: <z>%)
pause( (<p>,<q>,<w>) = richtige Antwort &Abbruch mit "Esc")
goto(start)

_falsch
.z = m * 100 / n
rou(z,0)
txv(25,<n> Versuche, <m> richtig . . . Ergebnis: <z>%)
pause( (<p>,<q>,<w>) = falsche Antwort &Abbruch mit "Esc")
goto(start)

end.
```

==========

Liste der 40 Demo-Skripts von PROGMATH.EXE
(+) = Skripts mit interaktiven Frage/Antwort-Eingaben
und nachfolgender Eingabe-Bewertung

==========

(1) Grundrechenarten I
(2) Grundrechenarten II
(3) Punkte und Strecken
(4) Kreise in der Ebene
(5) Rechtecke in der Ebene
(6) Dreiecke in der Ebene I
(7) Dreiecke in der Ebene II
(8) Dreiecke in der Ebene III
(9) Dreieck und Umkreis I
(10) Dreieck und Inkreis I
(11) Dreieck und Umkreis II
(12) Dreieck und Inkreis II
(13) Schlussrechnungen
(14) Rechteck und Umkreis
(15) Das kleine Ein-Mal-Eins (+)
(16) Einfache Gleichungen (+)
(17) Prozentrechnungen (+)
(18) Gleichungssysteme (+)
(19) Gleichschenkelige Trapeze (+)
(20) Zusammengesetzte Flächen (+)

---

(21) Division von Dezimalzahlen
(22) Quader im Schrägriss
(23) Pyramiden im Schrägriss
(24) Der Differenzialquotient
(25) Kurvendiskussion
(26) Das bestimmte Integral
(27) Die Bogenlänge von Kurven
(28) Inhalt und Schwerpunkt von Flächen
(29) Volumen und Schwerpunkt von Drehkörpern
(30) Mantelfläche von Drehkörpern
(31) Der Schwerpunkt von Kurven
(32) Rechnen mit komplexen Zahlen
(33) Quadratische Gleichungen
(34) Vektorrechnung in der Ebene
(35) Das Vektorprodukt
(36) Das Spatprodukt
(37) Kegelschnittstangenten I
(38) Kegelschnittstangenten II
(39) Kegelschnittsevoluten
(40) Die Ludolfsche Zahl "pi"

==========

Letzter Hinweis: Mit dem Passwort "free" wird eine Bibliothek von hunderten Mathematikskripts zur gesamten Schulmathematik geöffnet. Nach der Ausführung eines Skripts wird dieses immer automatisch in den Editor kopiert und kann dort eingesehen werden.

==========

E N D E