# FUNKTIONEN und PROGRAMME

# am

# TI-92

**T$^3$-SEMINAR - ST. PÖLTEN**

**23. April 1998**

**MAG. DR. THOMAS HIMMELBAUER**

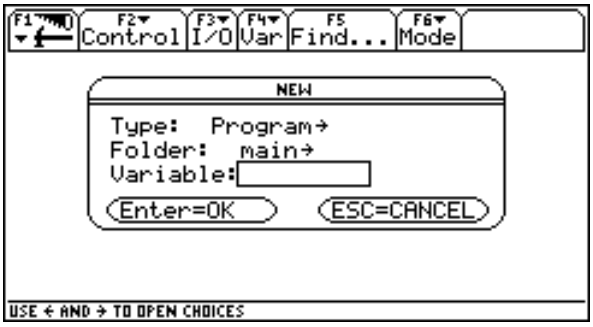
# SYSTEM VON 2 LINEAREN GLEICHUNGEN

Zunächst wollen wir eine Funktion definieren, die uns ein System von zwei linearen Gleichungen mit 2 Unbekannten löst.

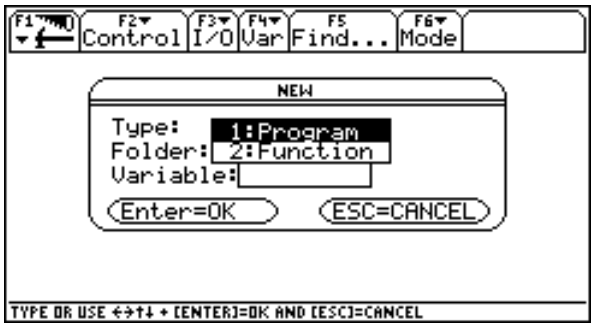
$$a_{11} \cdot x + a_{12} \cdot y = b_1$$

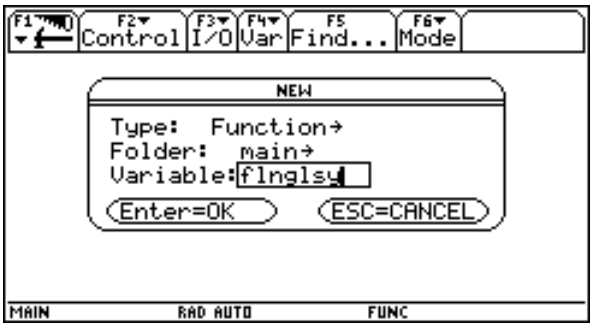
$$a_{21} \cdot x + a_{22} \cdot y = b_2$$

Im Programmeditor wählen wir New und stellen Type auf Function um.


Dann wählen wir den gewünschten Folder und geben der Function einen Namen.

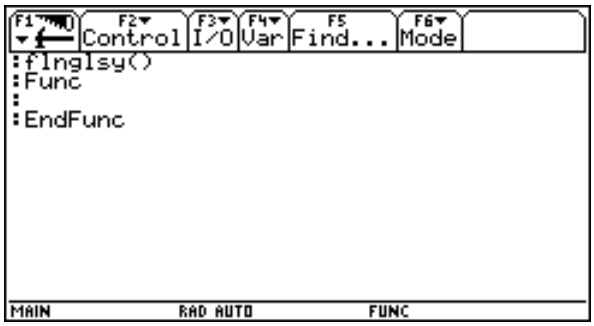


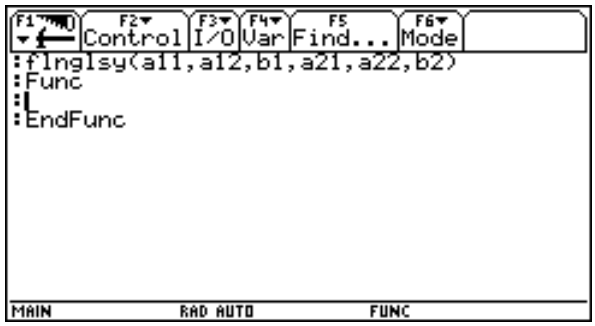




Es erscheint das Fenster zum Bearbeiten von Funktionen bzw. Programmen. Ganz oben steht der Name der Function flnglsy(), dann der Funktionsanfang Func und das Funktionsende EndFunc. Im Bereich zwischen Func und Endfunc müssen die von der Funktion durchzuführenden Befehle eingetragen werden. Jeder Befehl muß mit einem Doppelpunkt beginnen.
Die einzige Eingabemöglichkeit für Funktionen sind Parameter, die innerhalb der runden Klammern am Ende des Funktionsnamen angeführt werden. Sie werden durch ihre Reihenfolge voneinander unterschieden. Wir wählen als Eingabevariable natürlich die Koeffizienten und Konstanten unseres Systems. Beim Aufruf der Funktion muß die gewählte Reihenfolge der Variablen eingehalten werden.
z. B. führt der Aufruf flnglsy(3,4,7,6,-8,9) zur Lösung des Systems:

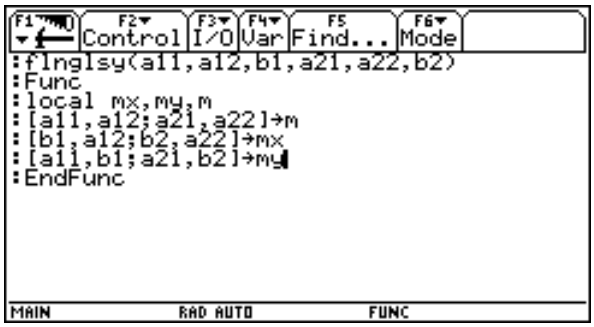
$$3x + 4y = 7$$

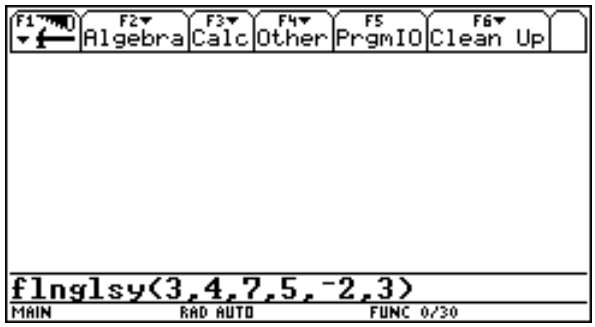
$$6x - 8y = 9$$



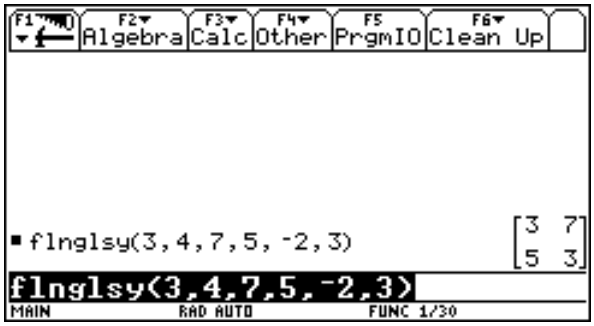



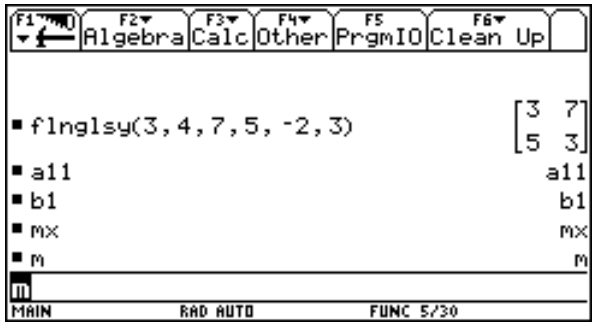

Eine Festlegung der Variablen nach ihrem Typ ist nicht notwendig. Das Gleichungssystem wollen wir nach der Cramerschen Regel lösen. Zunächst speichern wir die entsprechenden Matrizen unter den Variablen m, mx und my ab. Weil eine Funktion nur mit lokalen Variablen arbeiten kann, müssen diese drei Variablen als local angemeldet werden. Damit sind sie nur innerhalb der Funktion definiert und nach Ablauf der Funktion wieder unbelegt.



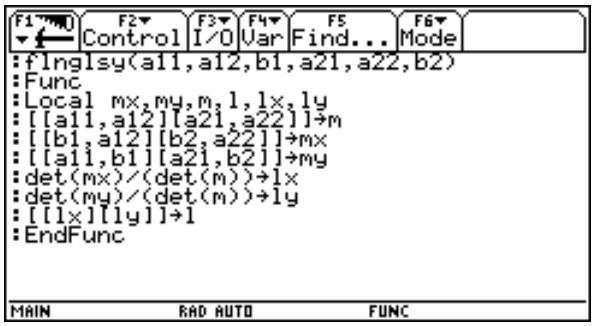

Mit 2nd Esc kehren wir in den Homebereich zurück und rufen unsere Funktion auf. Als Antwort bekommen wir das Ergebnis der letzten Befehlszeile. Wie wir nun überprüfen können, sind nicht nur die als local angemeldeten Variablen m, mx und my ohne Belegung, sondern auch die Eingabevariablen wie zum Beispiel a11. Sie werden immer local behandelt.

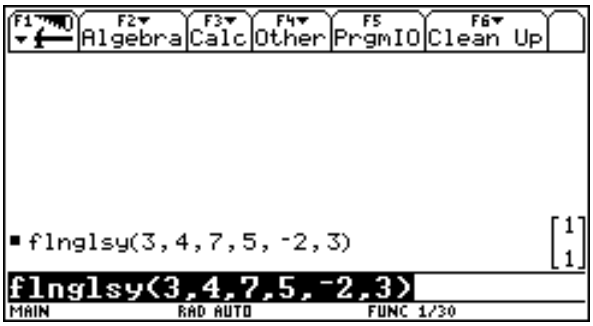


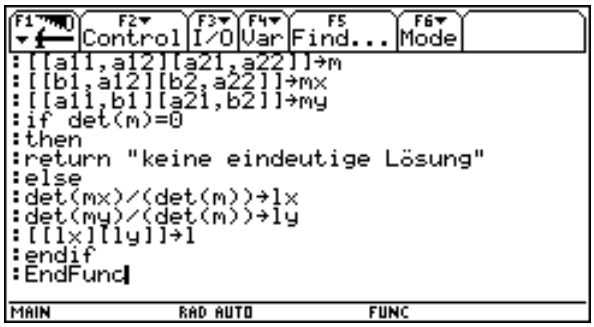

Über APPS/7/1 kehren wir in unsere Funktion zurück.

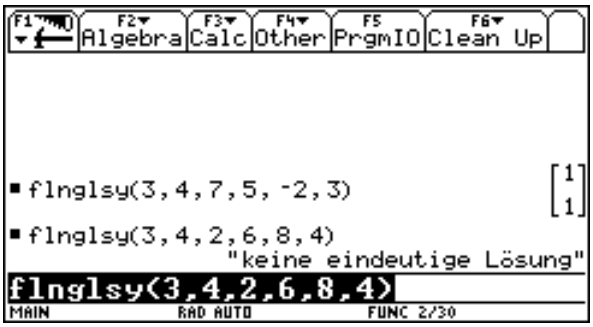
Es stellt sich nun die Frage, wie man mehrere Ergebnisse als Antwort erhalten kann. Die Lösung sind Listen oder Matrizen. Wir berechnen die beiden Lösungen lx und ly und bilden aus ihnen den Lösungsvektor l. Dabei dürfen wir nicht vergessen auch diese Variablen als local anzumelden. Dann können wir unsere Funktion wieder ausprobieren.




Da Divisionen durch Null nicht möglich sind, wollen wir nun mit einer Abfrage den Fall nicht eindeutig lösbarer Systeme ausschließen. Unter F2 Control findet man die entsprechenden Befehle. Um auch im Then-Zweig der Abfrage eine Antwort zu erzwingen muß der Befehl return verwendet werden. Die Anführungszeichen dienen zur Kennzeichnung von Text. Nun können wir die Funktion für beide Fälle testen.

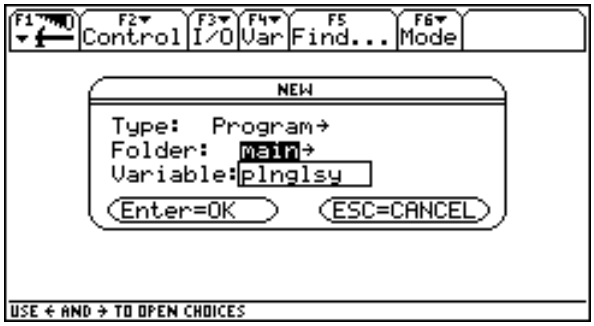


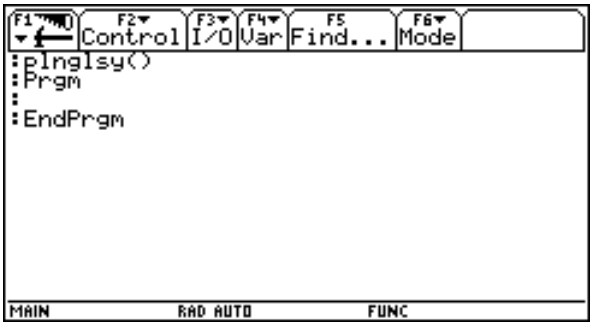




Den Schülern bereitet es kaum Probleme Funktionen ähnlicher Art zu definieren und zu verwenden. Es ist daher weder sinnvoll noch auf die Dauer möglich, das Verwenden solcher selbst definierten Funktionen zu verhindern.

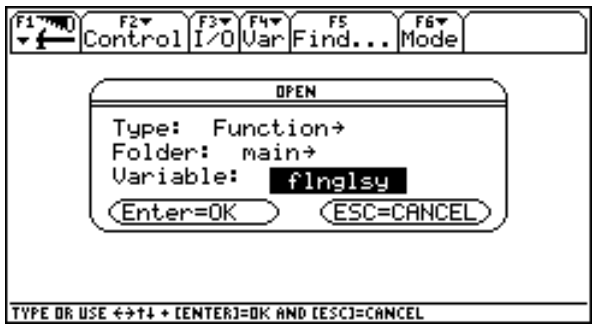
Nun wollen wir das gleiche Problem mit einem Programm lösen. Das hat den Vorteil, daß eine Vielzahl von Anweisungen zur Verfügung stehen, die praktisch alle Bereiche des Rechners steuern können, den Nachteil, daß die Ausgabe auf dem Input-Output-Schirm und nicht im Homebereich erfolgt.

Wir öffnen den Programmeditor, lassen als Type Programm und legen einen Programmnamen fest. Es erscheint eine zu den Funktionen analoge Struktur.

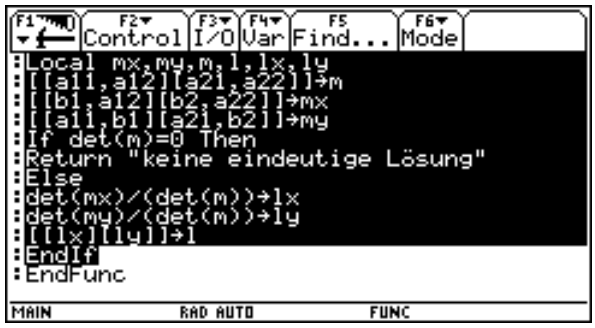


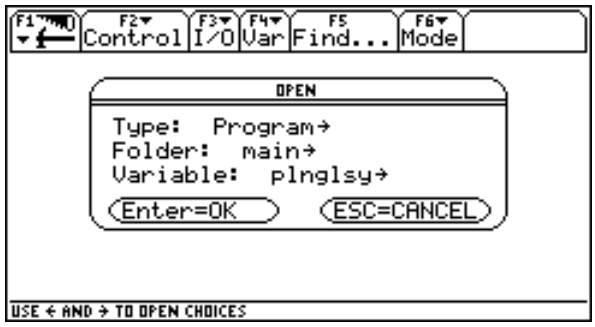
Um nicht alle Befehle neu eingeben zu müssen, öffnen wir unsere Funktion.

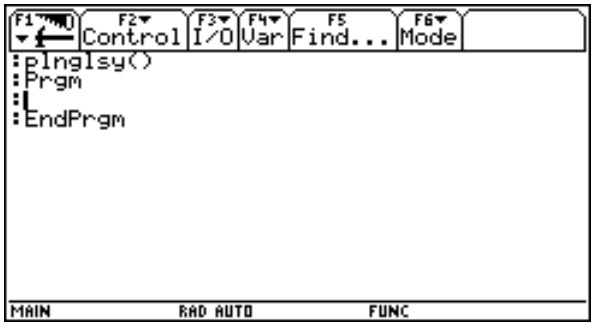

Wir markieren die Befehlszeilen (SHIFT und Cursor-Pad) und kopieren sie in die Zwischenablage (Diamant C).

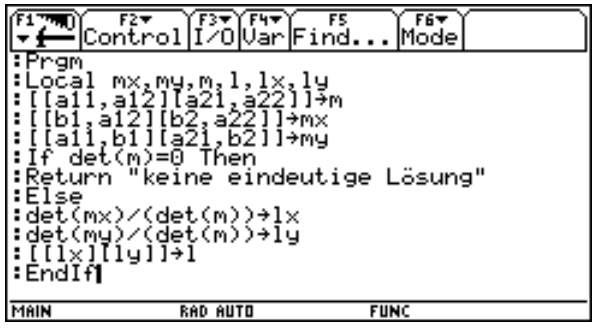




Dann kehren wir in unser Programm zurück.

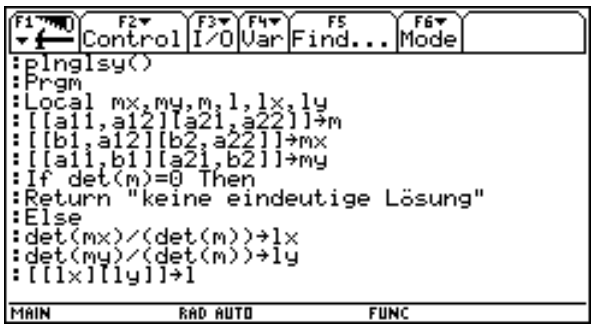



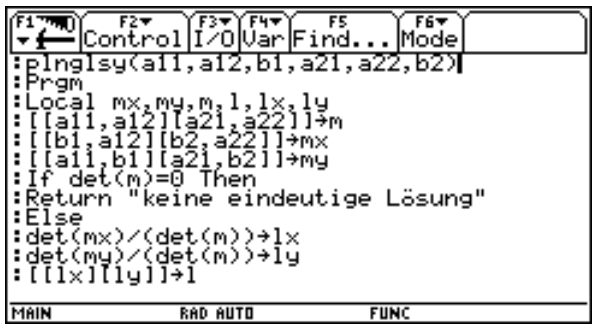

Hier fügen wir die Befehlszeilen ein (Diamant V).



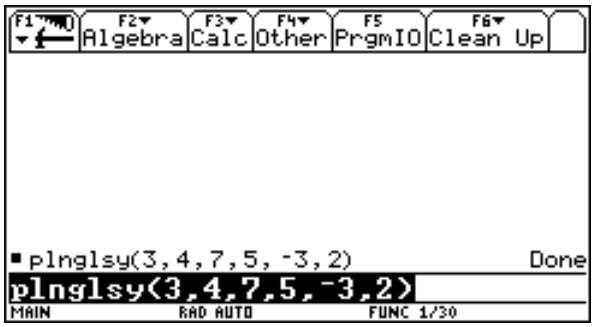



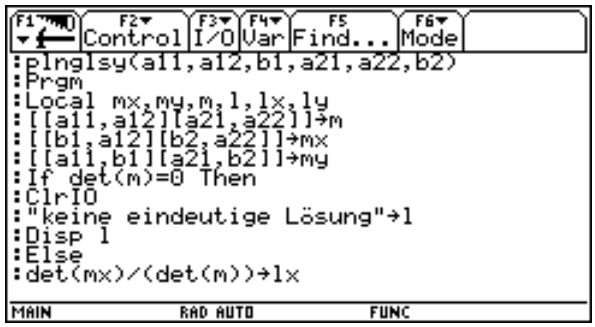
Dann fügen wir die Eingabevariablen ein.



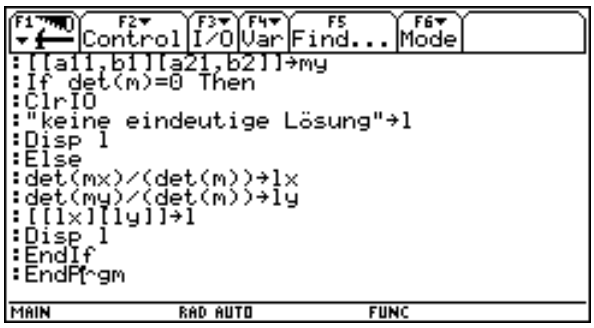



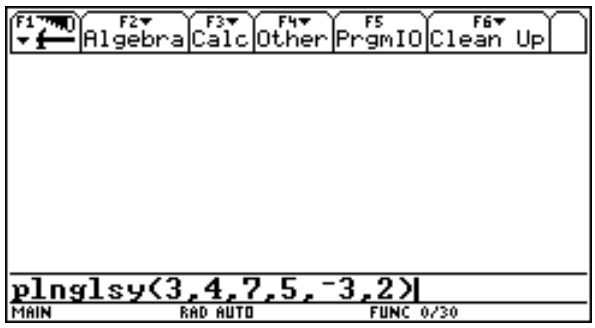

Zurück im Homebereich testen wir das Programm. Die Antwort besteht nur aus der Meldung “done”. Ein Programm benötigt eigene Ausgabebefehle. Einer davon ist disp. Er gibt in einer neuen Zeile des IO-Schirmes die Belegung der nachfolgenden Variblen aus. Wenn früher ausgegebene Inhalte nicht erwünscht sind, muß der IO-Schirm zuvor mit ClrIO gelöscht werden.



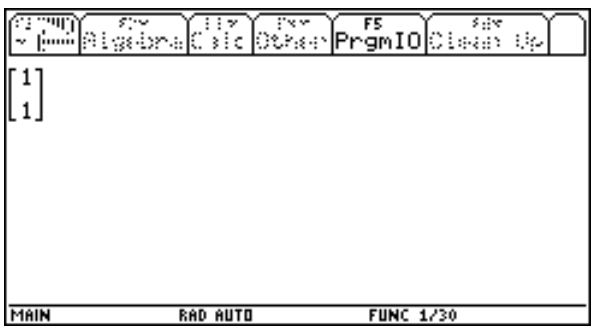



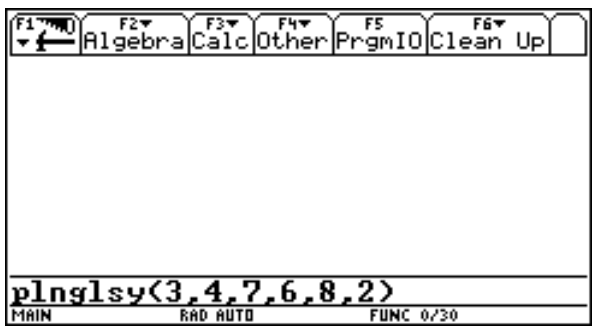

Dann können wir unser Programm testen.



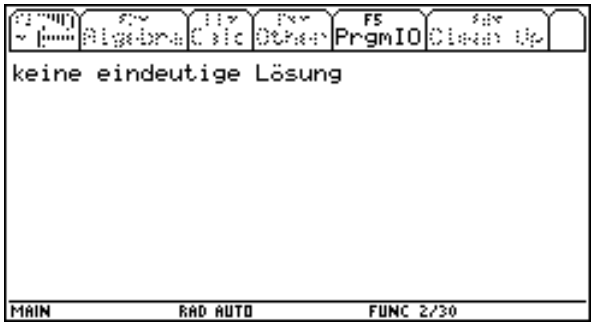



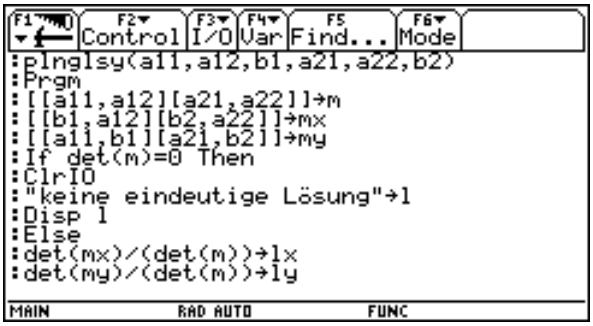

Vom Homebereich wird in den IO-Schirm gewechselt und die Lösung angezeigt. Über F5 kommen wir wieder in den Homebereich zurück und testen auch noch den anderen Ausgabefall.



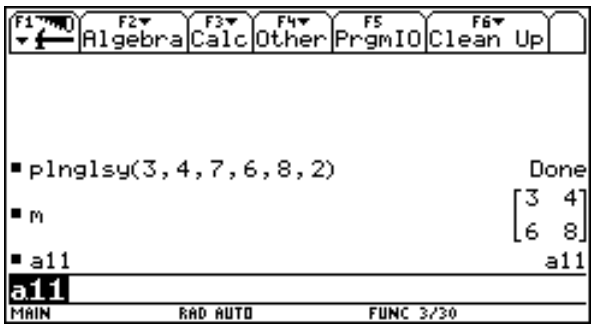



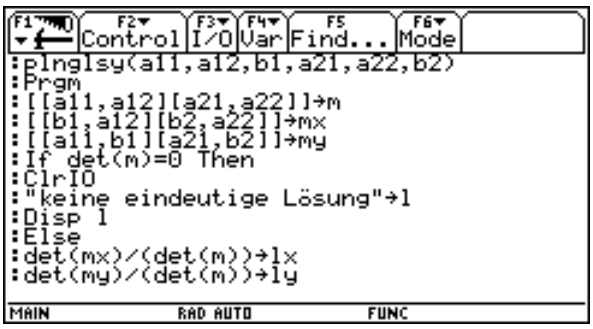
Nun löschen wir die Definition der localen Variablen aus dem Programm.



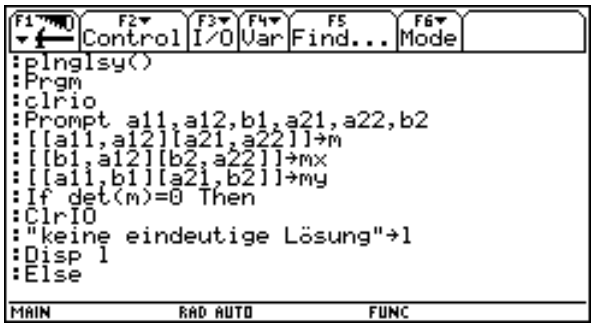



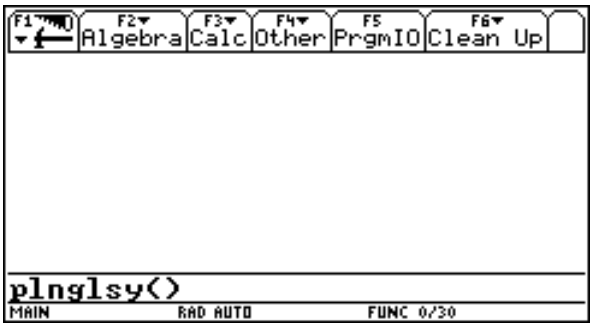

Ein Programm läuft auch mit nicht localen Variablen. Die Variablen des Programmes sind nun auch nach Ablauf des Programmes belegt. Nur die Eingabevariablen bleiben automatisch local.
In der Regel ist es günstiger nur mit localen Variablen zu arbeiten, um später nicht von der Belegung mancher Variablen überrascht zu sein. Außerdem verbrauchen die globalen Variablen Speicher.



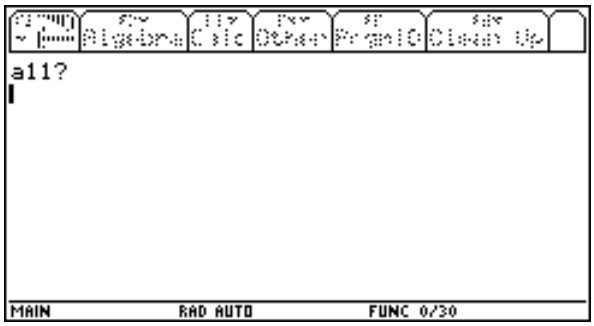



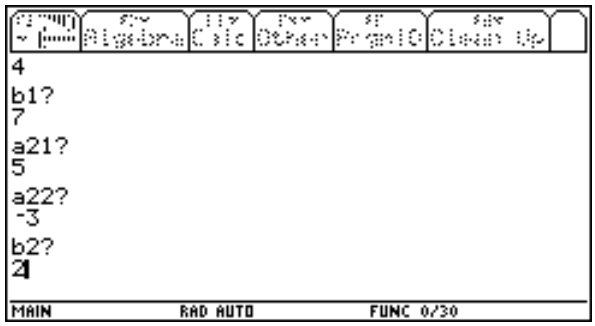

Eine weitere Eingabemöglichkeit stellt der Promptbefehl dar. Wir übergeben dem Programm keine Eingabevariablen. Die runden Klammern müssen aber erhalten bleiben. Nach einem Löschen des IO-Schirmes rufen wir über Prompt die einzugebenden Variablen auf. Sind sie nicht als local festgelegt, so sind sie global. Dann testen wir das Programm. Der Aufruf muß mit den runden Klammern erfolgen.

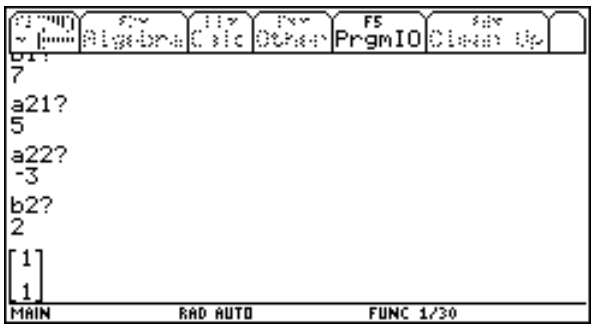


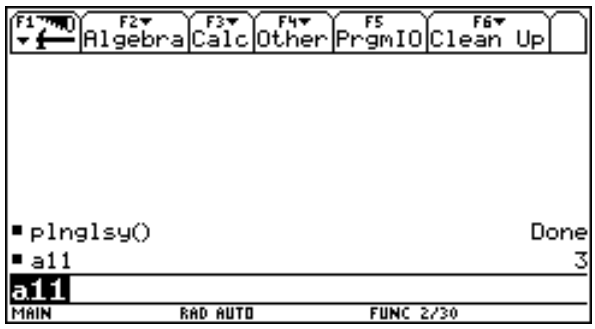




Es meldet sich der IO-Schirm mit der im Promptbefehl zuerst angeführten Variablen a11 mit einem Fragezeichen dahinter. Der Cursor steht eine Zeile tiefer. Wir geben den Wert für a11 ein, bestätigen mit Enter, es erscheint a12 mit Fragezeichen usw.




Da wir die Variablen nicht als local definiert haben, ist z. B. a11 auch nach Ablauf des Programmes mit dem eingegebenen Wert belegt.

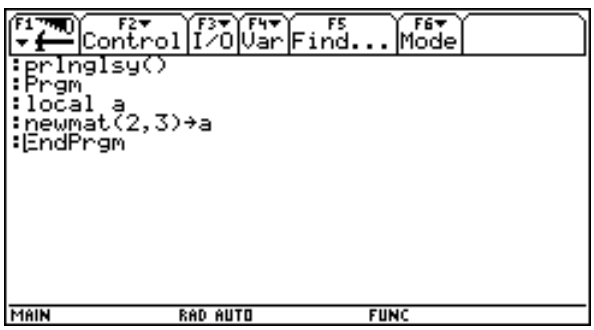
Es gibt nun aber noch weitere wesentlich aufwendigere, aber auch übersichtlichere und benutzerfreundlichere Eingabemöglichkeiten.

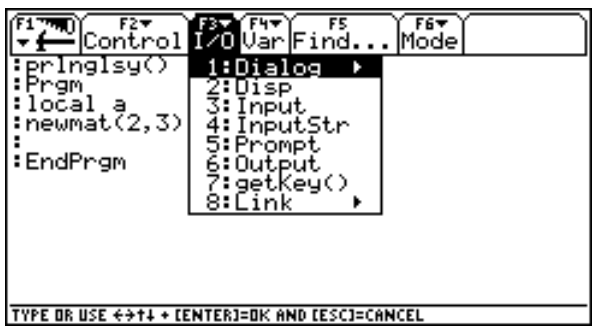
Dazu wollen wir im linearen Gleichungssystem eine Umbezeichnung vornehmen.

$$a_{11} \cdot x + a_{12} \cdot y = a_{13}$$

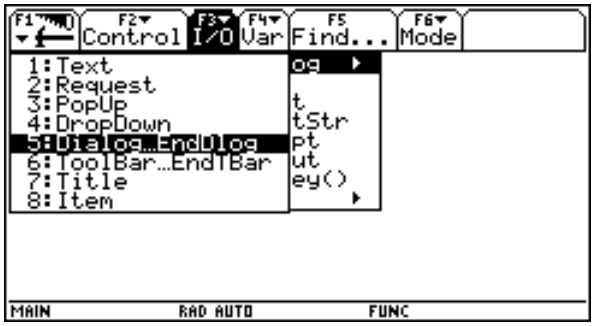
$$a_{21} \cdot x + a_{22} \cdot y = a_{23}$$

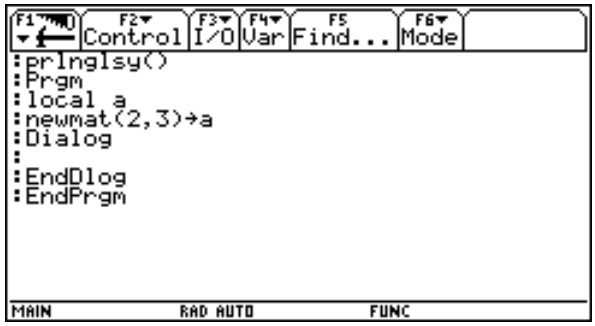
Wir öffnen ein neues Programm, defineren die Variable a als local und weisen ihr eine leere (mit lauter Nullen besetzte) 2x3 Matrix zu. Danach wollen wir eine Dialogbox für die Eingabe aufrufen. Die entsprechenden Befehle finden wie unter F3/Dialog.



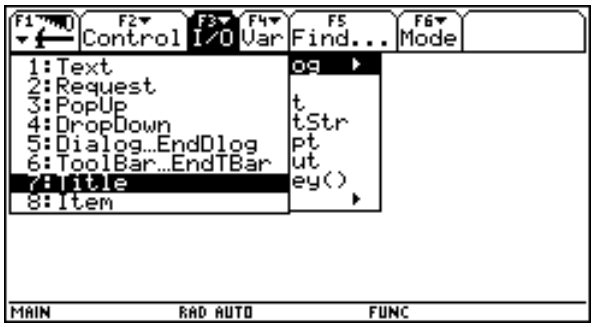



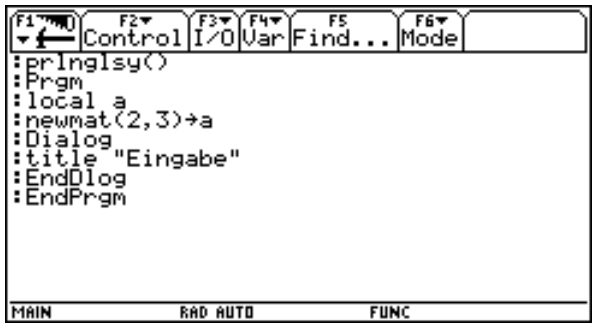
Wir fügen die Befehle für Anfang und Ende des Aufrufes einer Dialogbox ein.



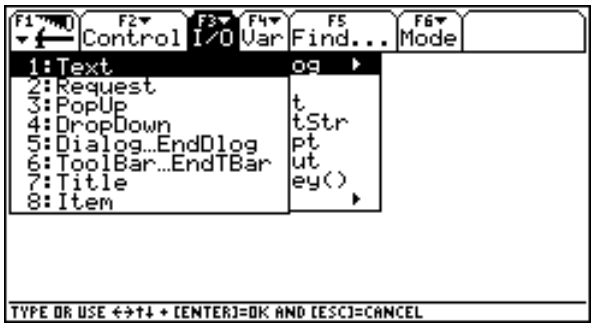



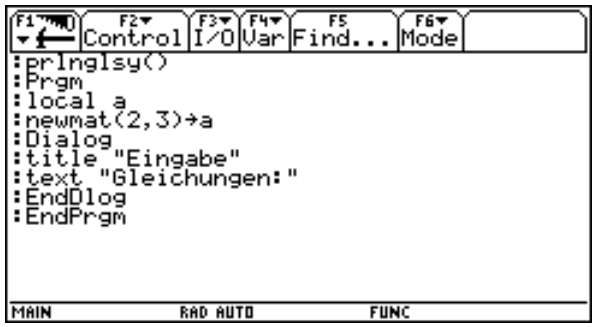

Danach holen wir uns den Befehl Title, um die Überschrift der Dialogbox festzulegen. Wir wählen “Eingabe”.

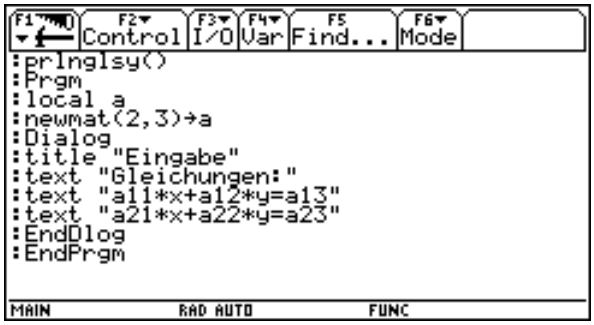


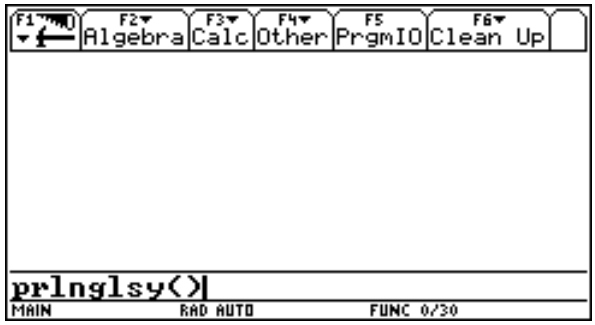




Zur Information des Benutzers wollen wir einige Erläuterungen anführen. Dazu dient der Befehl Text. Wir geben die Gleichungen in der von uns gewählten Bezeichnungsart an.

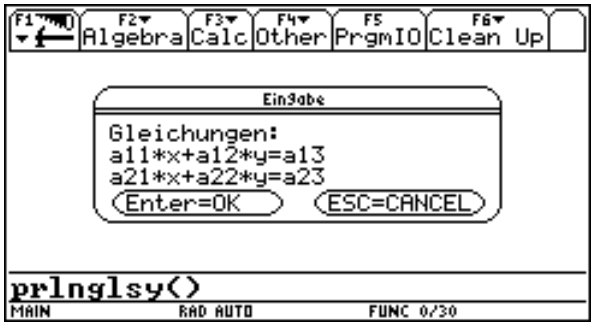


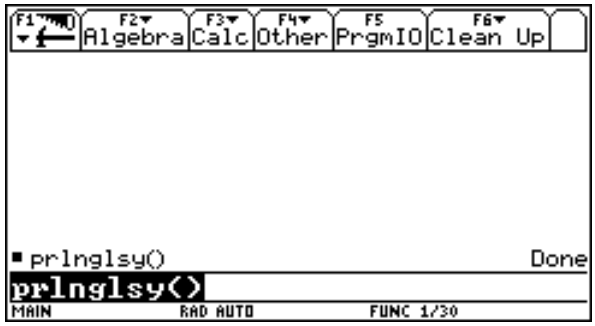



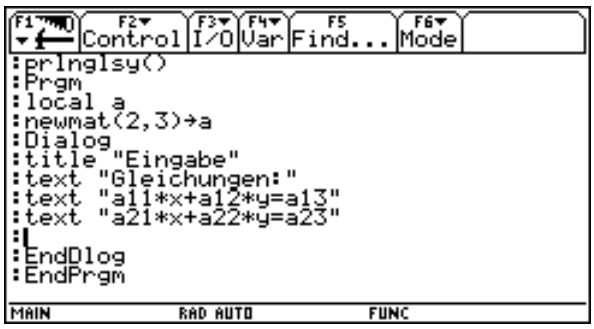



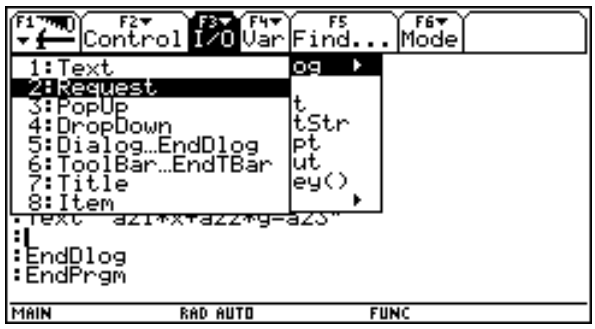
Jetzt können wir die Dialogbox bereits ausprobieren. Mit Enter oder ESC wird die Box geschlossen und das Programm beendet.



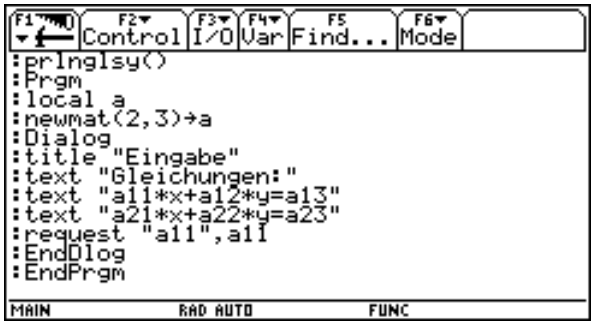



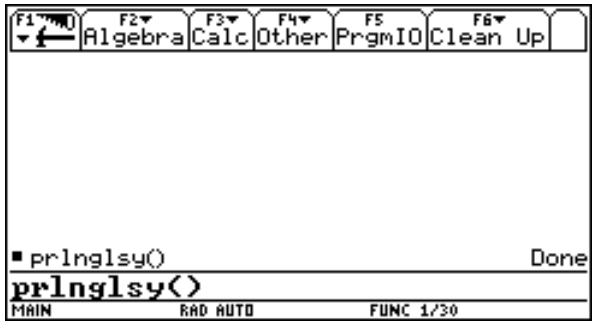

Der Befehl, der eine Eingabe fordert, lautet Request. Ihm folgt ein Text und die Variable unter der die Eingabe als Text abgespeichert wird.



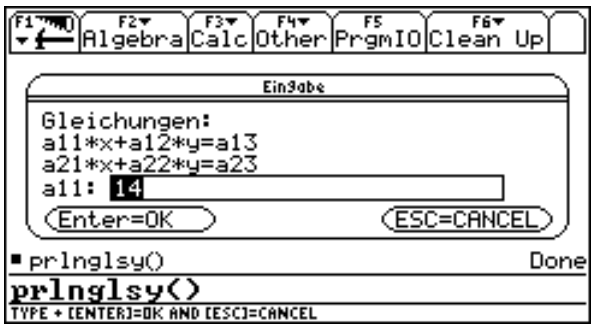



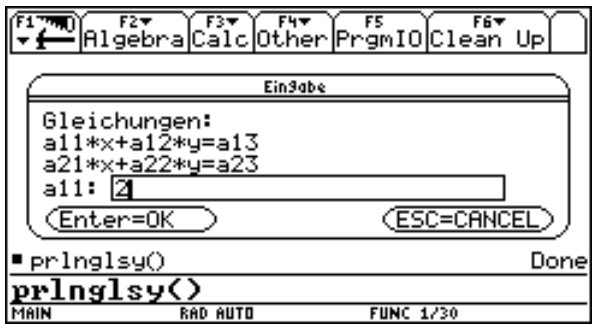

Daher lautet unsere Befehlsezeile: request “a11”, a11



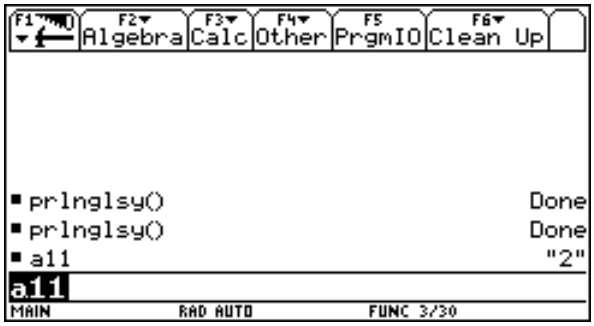



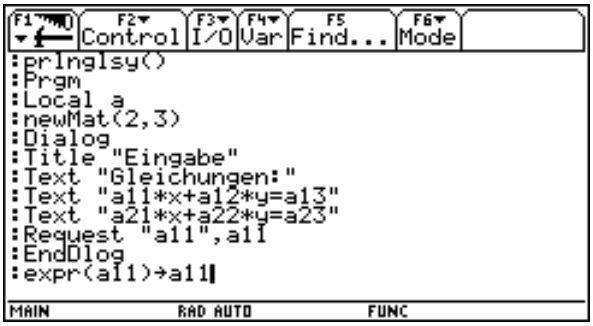

Probieren wir die veränderte Box gleich aus. Es erscheint eine zusätzliche Zeile mit dem Text a11 und einem Feld mit Eingabemöglichkeit. Die angezeigte Zahl ist durch die Belegung der Variablen a11 verursacht. Wir verändern auf 2 und bestätigen mit doppeltem Enter. Die Box schließt und das Programm ist beendet.



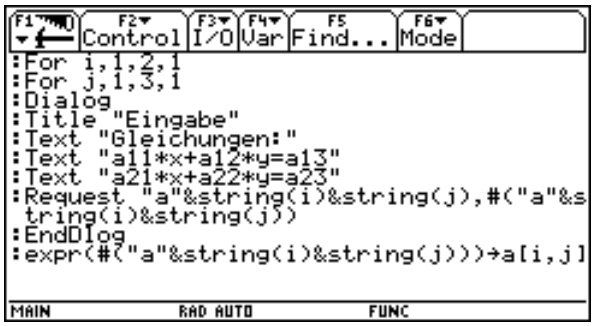



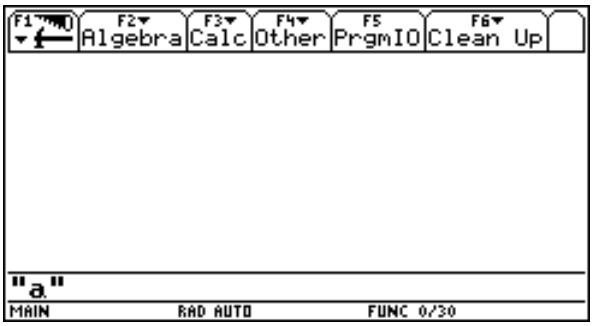
Wir können die neue Belegung von a11 überprüfen. Wie schon gesagt wurde die Zahl 2 als Text abgespeichert. Um diesen Text in eine Zahl zu verwandeln fügen wir im Programm den Befehl expr ein.



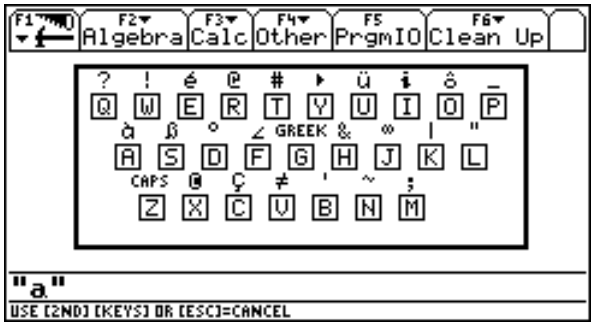



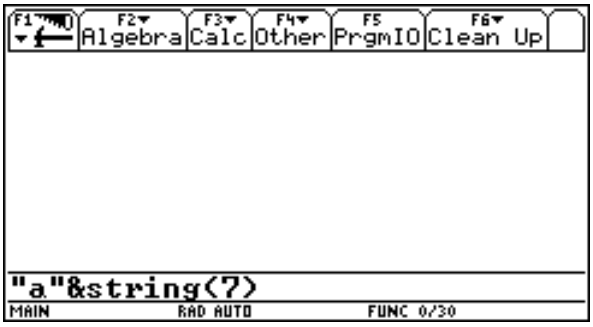

Um nicht 6 Dialogboxen programmieren zu müssen, bauen wir nun eine Doppelschleife in unser Programm ein. Die Zahl i läuft von 1 bis 2 und die Zahl j für jeden der beiden Fälle von i von 1 bis 3. Damit sind alle Indizes aus unseren Gleichungen und alle Zeilen- und Spaltenbezeichnungen unserer Matrix a abgedeckt. Nun müssen wir die beiden Befehle request und expr in Abhängigkeit von i und j programmieren. Sie sehen unterhalb schon die fertigen Befehle. Wir wollen sie nun schrittweise erklären. Dazu gehen wir in den Homebereich und geben den Buchstaben a als Text ein.



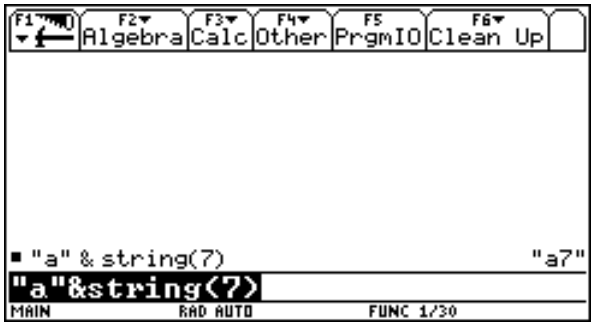



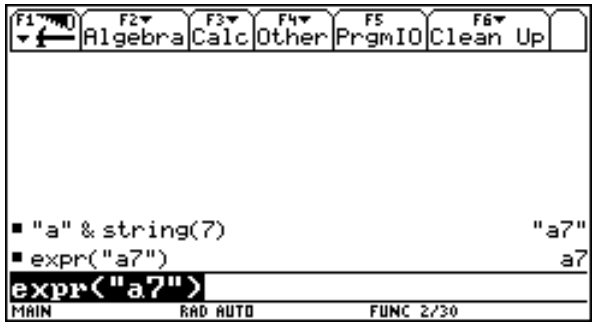

Um an diesen Text ein anderes Textstück anzuhängen gibt es ein eigenen Operator. Wir finden ihn über Diamant K als 2nd-Funktion von H. Und fügen den Text bestehend aus der Ziffer 7 an. Der Befehl string verwandelt uns einen Term in einen Text.



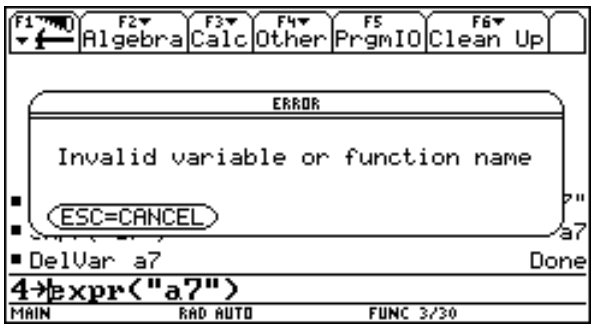



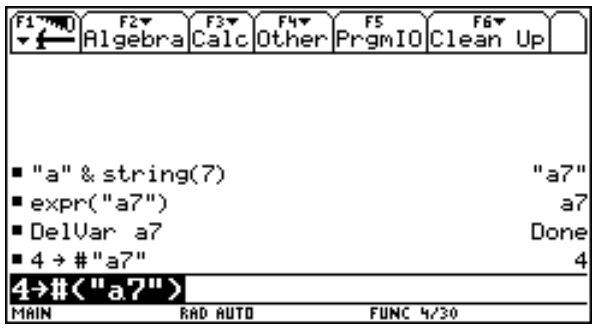
Die Antwort ist der Text a7, der mit Hilfe von expr in den Term a7 verwandelt werden kann.



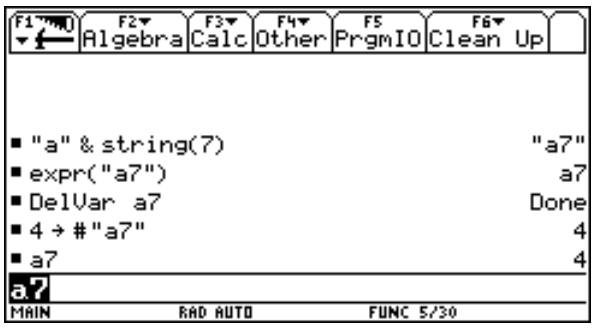



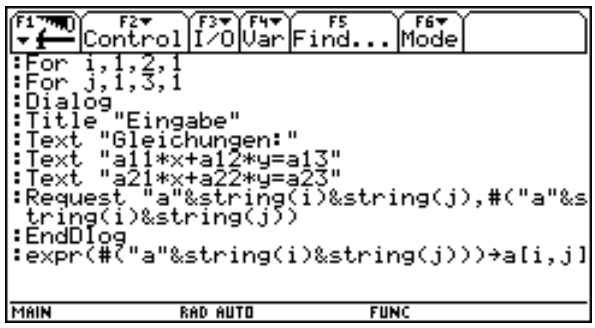

Nun aber Vorsicht! Während man a7 mit einer Zahl belegen kann, kann man expr(“a7”) mit keiner Zahl belegen. Denn expr(“a7”) wird als Term und nicht als Variable erkannt. Um einen Text in eine Variable zu verwandeln gibt es eine eigenen Operator. Wir finden ihn als 2nd-Funktion von T.



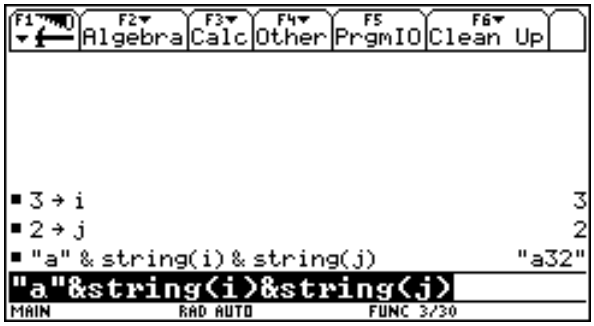



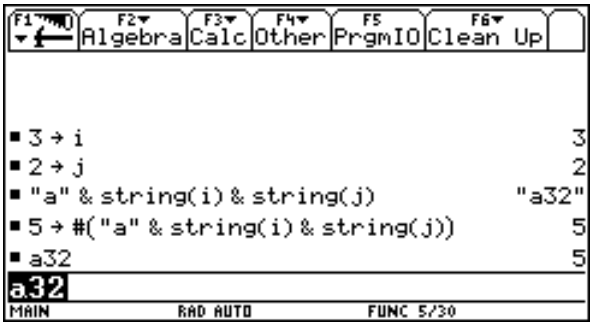

Daher bildet also “a”&string(i)&string(j) den Text aij und #( “a”&string(i)&string(j)) bildet die Variable aij.
Die unter der Variablen aij (#( “a”&string(i)&string(j))) als Text abgespeicherte Zahl wird mit Hilfe von expr (#( “a”&string(i)&string(j))) in eine Zahl verwandelt und an die i-te Zeile und j-Spalte der Matrix abgespeichert.



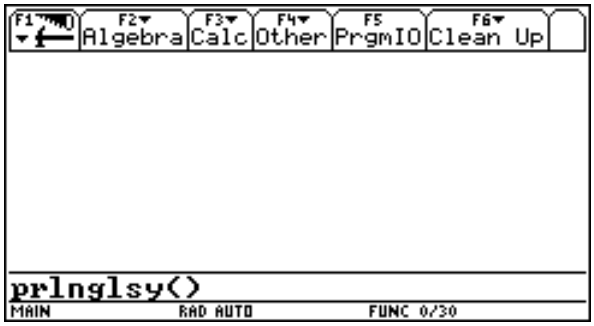



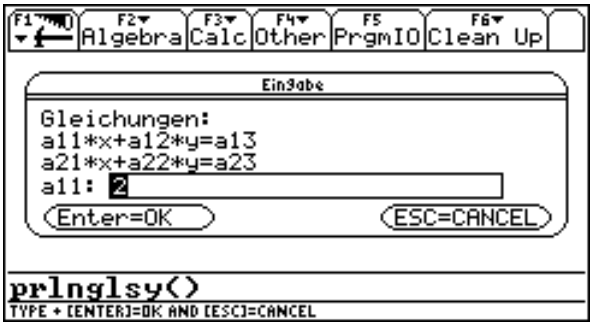
Diesen Vorgang können wir uns für eine Belegung von i und j im Homebereich veranschaulichen.

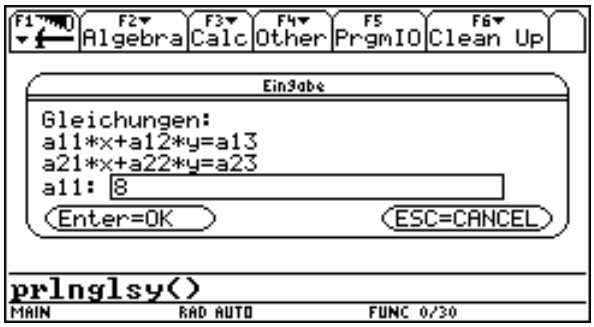


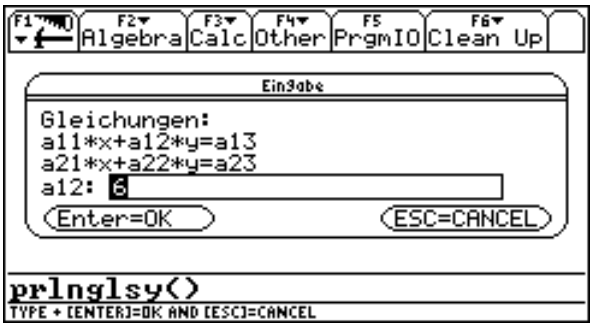



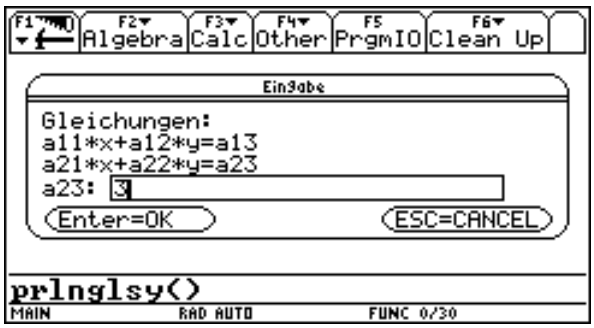

Nun können wir unsere Dialogboxen testen

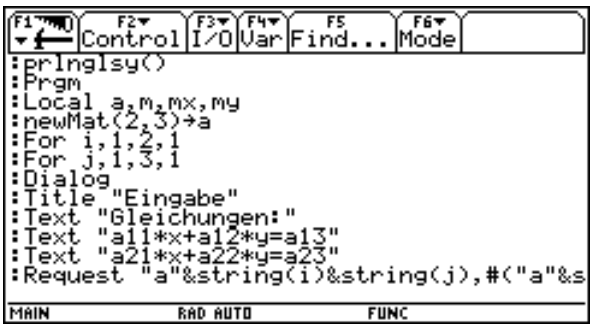


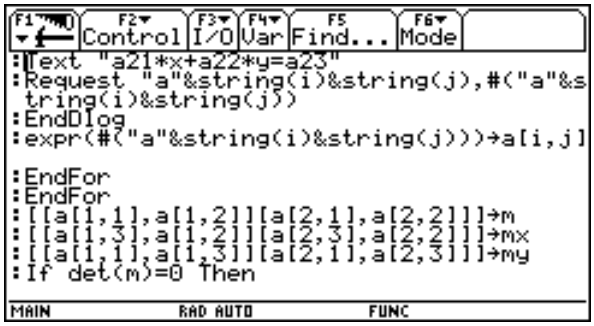



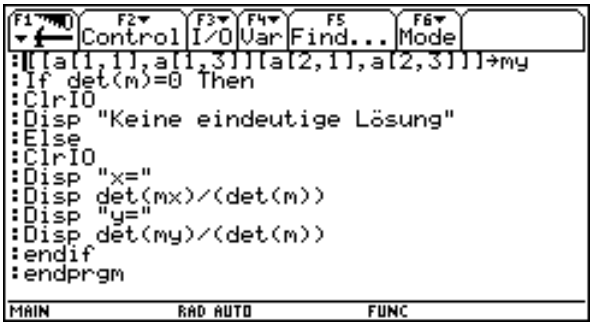



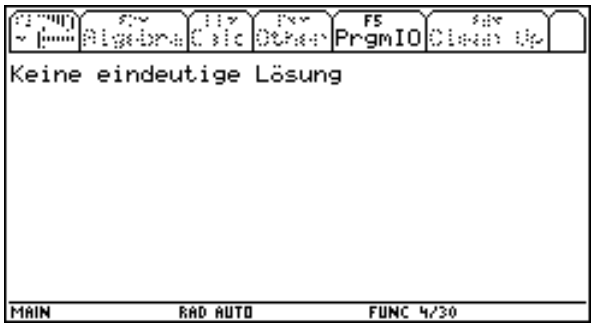



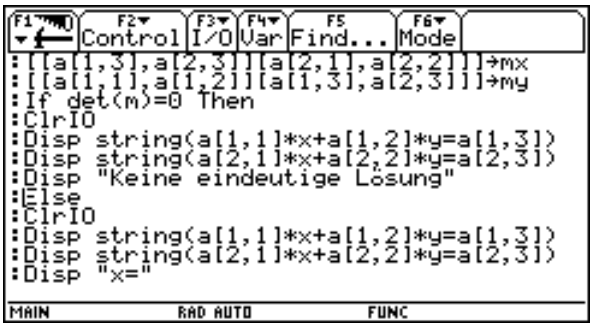
Wir setzen mit der schon bekannte Programmierung der Cramerschen Regel fort.



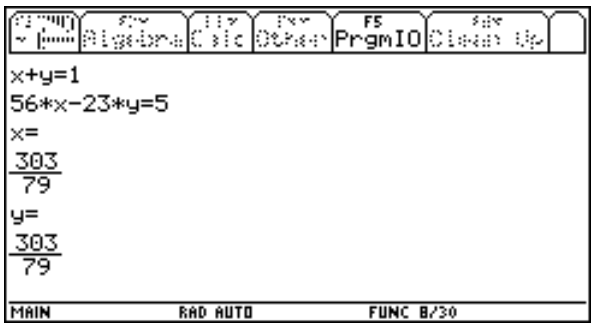



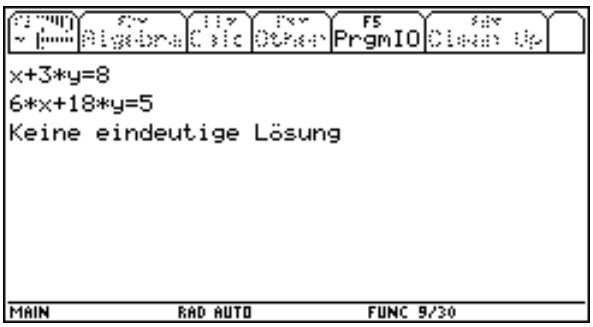

Um Eingabefehler zu vermeiden ist es günstig bei der Ausgabe die eingegebenen Gleichungen mit anzuführen.



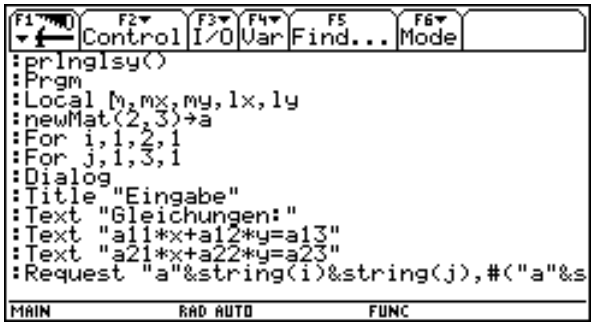

Nun können wir das Programm testen und erhalten je nach Art des Gleichungssystem die folgenden Ausgaben.

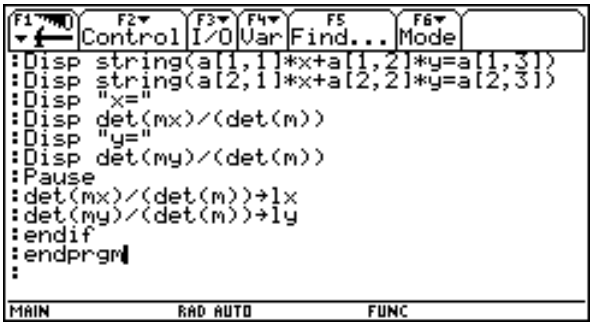


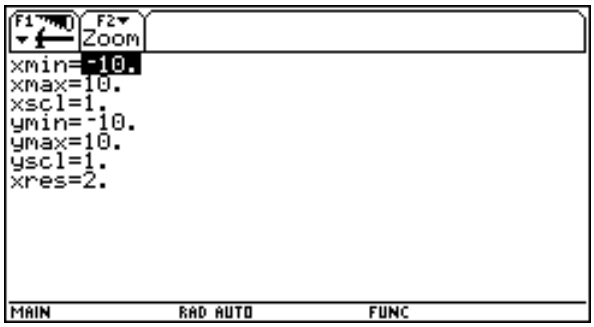

Da auch der Graphikbereich und der Y-Editor von Programmen gesteuert werden können, wollen wir nun für den Fall einer eindeutigen Lösung die durch die Gleichungen bestimmten Geraden zeichnen lassen. Dazu kehren wir in das Programm zurück und fügen am Ende unserer bisherigen Berechnung eine Pause ein, damit wir unsere Ausgabe der Lösungen beliebig lange betrachten können.

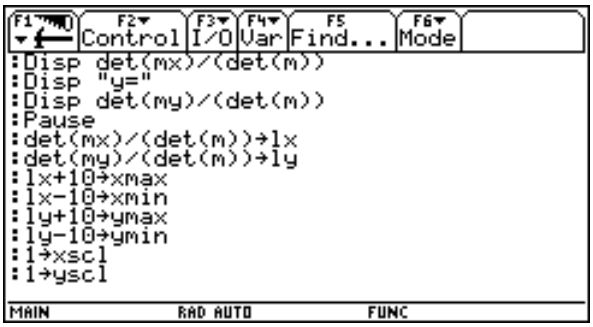
Danach weisen wir die Lösung für x und y den Variablen lx und ly zu.



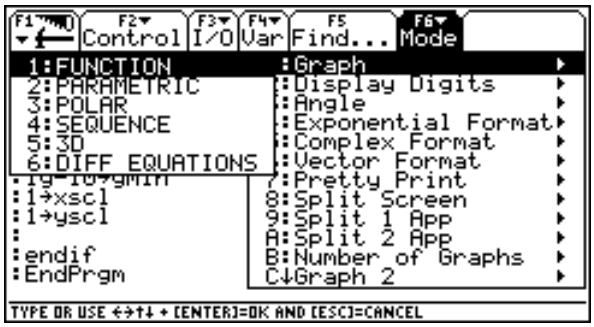



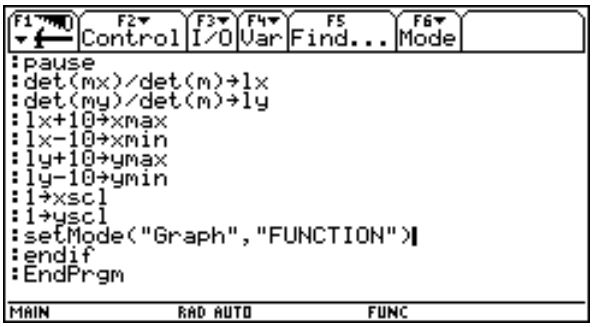

Wir wollen nun die Windowvariablen so einstellen, daß der Schnittpunkt der beiden Geraden im Bildmittelpunkt zu liegen kommt. Um uns an die Windowvariablen zu erinnern, öffnen wir die Windoweinstellung. Um für den "normalen Hausgebrauch" eine vernünftige Einstellung zu bekommen, erstrecken wir das Fenster symmetrisch um den Schnittpunkt in jeder Achsenrichtung 10 Einheiten groß.



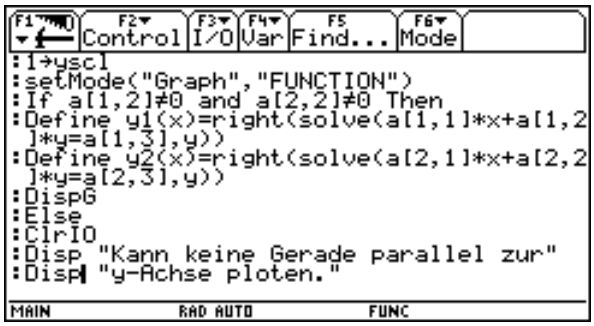



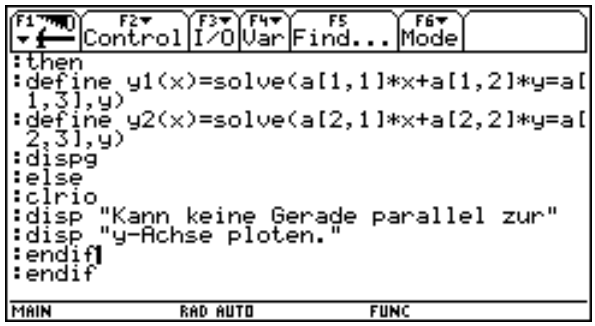

Wir stellen den Graphikmode auf Function um. Entsprechende Befehle finden wir bei F6 Mode.



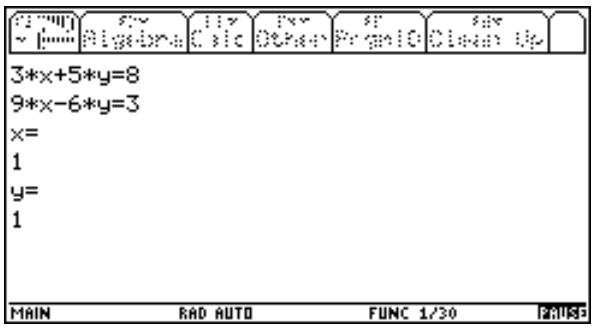



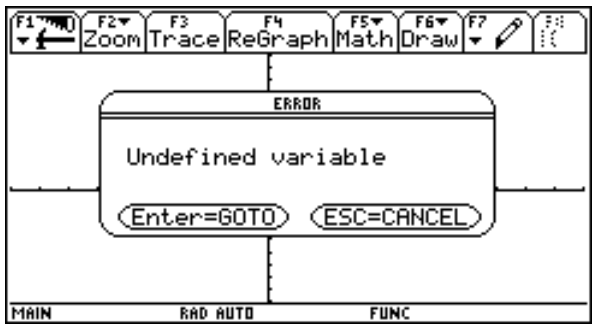

Da sich in dieser Graphikart zur y-Achse parallele Geraden nicht zeichnen lassen, müssen wir diese Fälle (a12 = 0 oder a22 = 0) mit einer Abfrage ausschließen. Danach müssen wir die entsprechenden Funktionen für den y-Editor definieren. Dazu lösen wir die Gleichungen mit solve nach y auf und weisen die rechten Gleichungsseiten mit dem Befehl right den Funktionen y1 und y2 des Editors zu.
Für den Fall zur y-Achse paralleler Geraden, geben wir aus, daß solche Geraden nicht gezeichnet werden können.



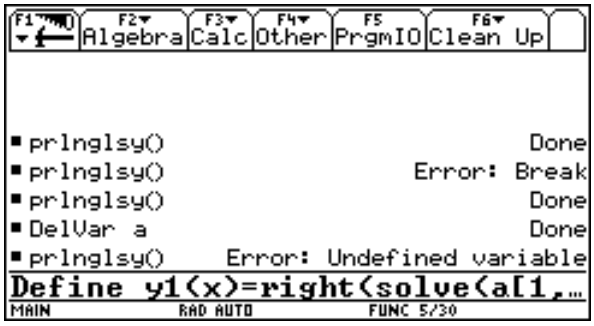



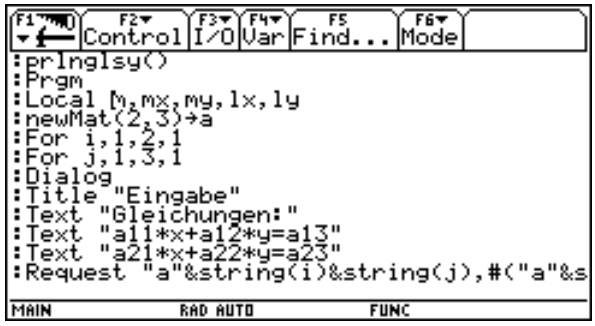

Nun können wir das Programm wieder testen. Zunächst erscheint der IO-Schirm mit der bekannten Ausgabe der Gleichungen und Lösungen und das Pausezeichen ist rechts unten zu sehen. Mit Enter setzt das Programm seine Arbeit fort, hält aber sofort mit folgender Meldung an.



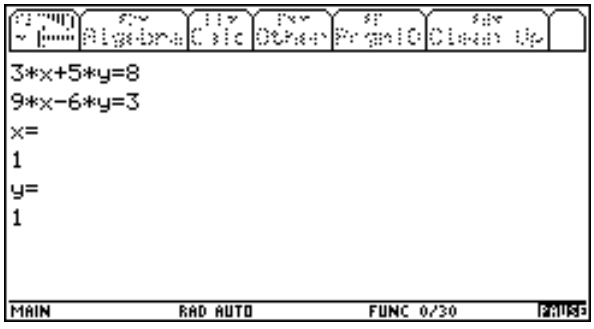



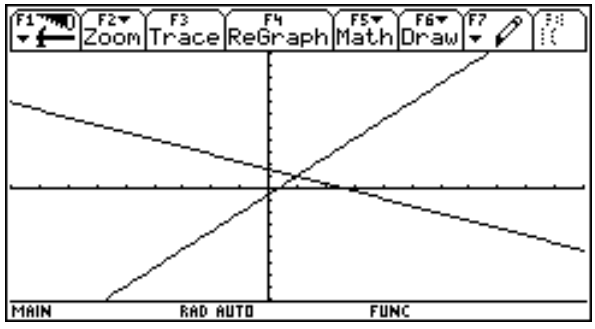

Diese wird durch Enter genauer beschrieben. Die Funktionen im y-Editor können nur auf globale Variablen zugreifen. Daher müssen wir die Matirix a aus der Liste der localen Variablen streichen.



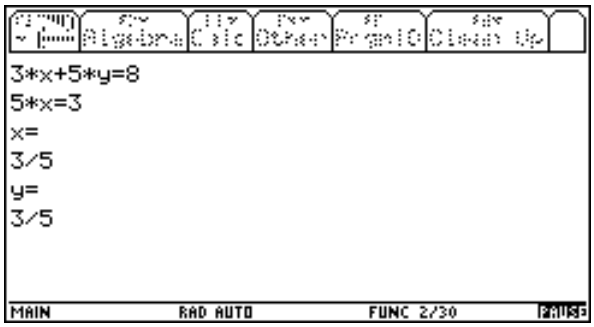




Nun erhalten wir bei erneutem Testen das gewünschte Resultat.



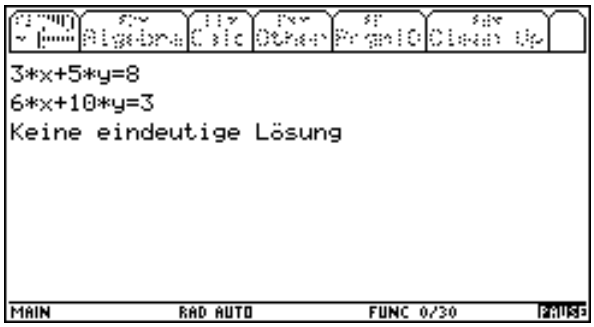



Bei entsprechend anders gewählten Gleichungssystemen erhalten wir für den Fall, daß eine der Geraden parallel zur y-Achse verläuft.

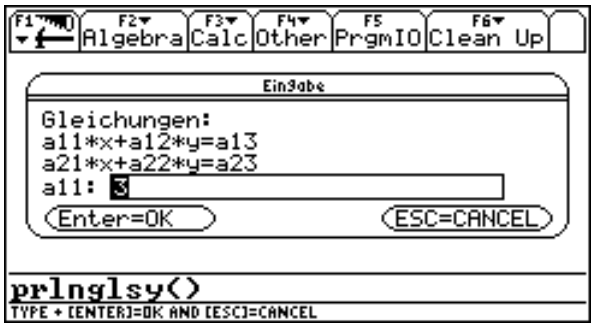


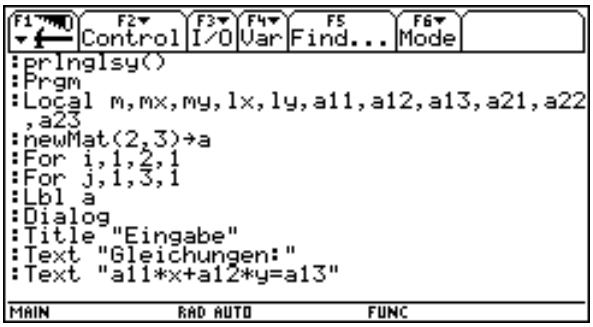


Und folgende Ausgabe, wenn keine eindeutige Lösung besteht.

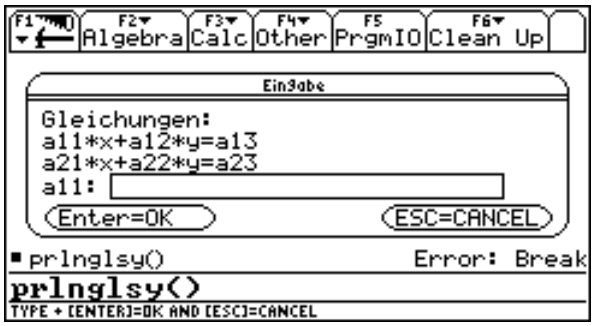



Jetzt wollen wir uns weiteren Verbesserungen zuwenden. Die in der Dialogbox für die aij voreingestellten Werte können wir dadurch entfernen, daß wir diese Variablen als local erklären. Damit sind sie bei Aufruf der Box unbelegt.



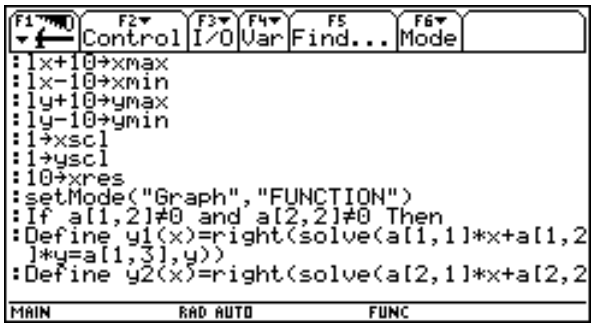



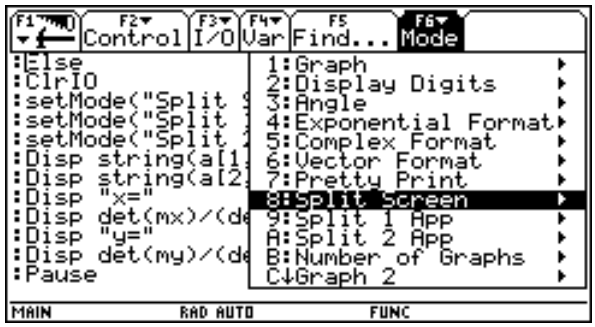

Weiters wollen wir verhindern, daß die Box durch Drücken von Enter oder Esc verlassen wird, bevor noch eine Eingabe gemacht wurde. Das Verlassen der Box durch ESC wird durch die Systemvariable ok überwacht. Sie erhält dann den Wert Null. Das Verlassen der Box mit Enter, ohne eine Eingabe gemacht zu haben, können wir daran erkennen, daß der unter aij gespeicherte Text keinen Buchstaben enthält. Dies können wir mit dem Befehl dim abfragen. Dieser gibt im Falle eines Textes, der keinen Buchstaben enthält, den Wert Null. Wir führen eine entsprechende Abfrage durch und springen mit dem goto Befehl über den Lbl a wieder zum Aufruf der entsprechenden Box zurück.



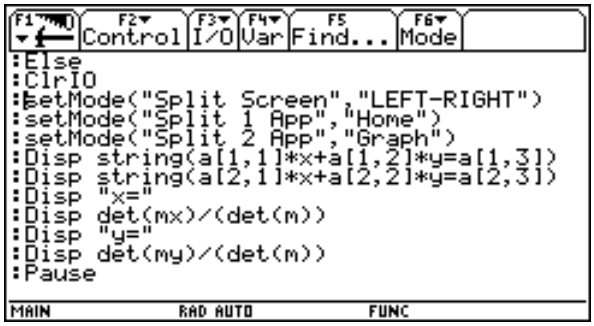


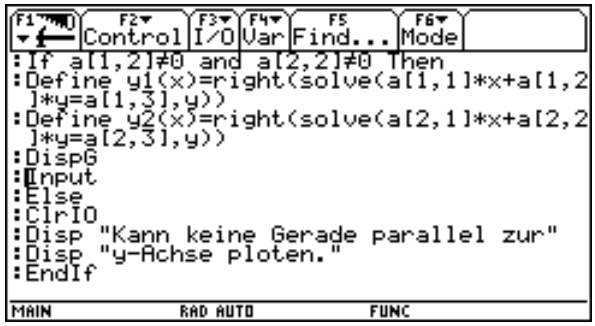
Aussteigen aus dem Programmablauf kann man nach wie vor über ON.
Dann kehren wir im Programm zur Graphikeinstellung zurück und setzten xres auf 10, um das Tempo des Zeichnens der Geraden zu erhöhen. Hernach wollen wir den Bildschirm rechts-links-teilen, um links den IO-Schirm und rechts die Graphik sehen zu können. Die entsprechenden Befehle finden wir bei F5 Mode.



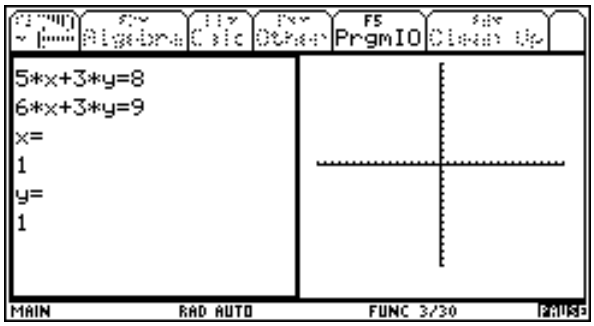



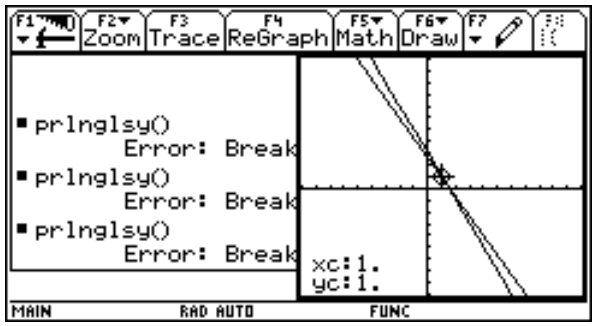

Hinter dem Befehl dispG, der uns den Graphikschirm zeigt, fügen wir den Befehl input ein. Dadurch erscheint der Graphicursor in der Mitte des Graphikfensters, also in unserer Einstellung auf dem Schnittpunkt. Mit Enter werden dann die Koordinaten des Graphikcursors in die Systemvariablen xc und yc abgespeichert. Also sind in unserem Falle dann xc und yc mit der Lösung des System belegt.



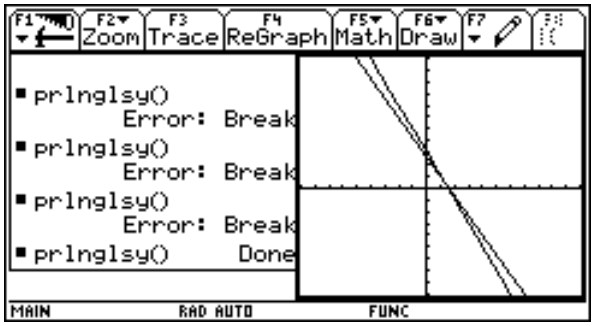



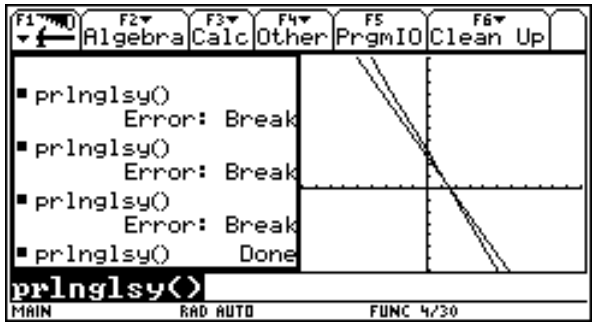

Nun testen wir wieder für ein System mit einer eindeutigen Lösung. Es erscheint der geteilte Bildschirm, der linke Teil ist aktiv und zeigt den IO-Schirm. Mit Enter wird der rechte Teil aktiv, die Graphik wird gezeichnet, der Graphikcursor erscheint am Schnittpunkt. Mit Enter werden seine Koordinaten abgespeichert.



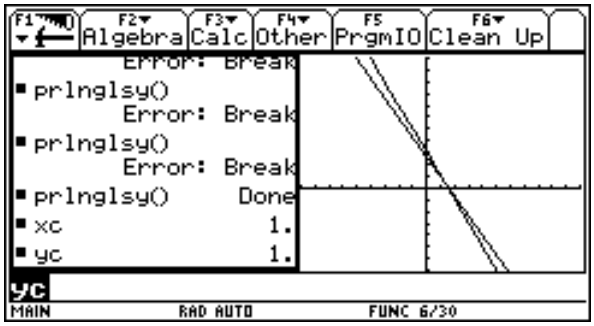



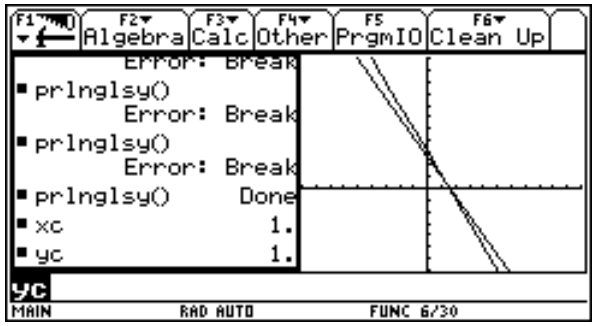

Der Graphikcursor verschwindet. Wir wechseln in den linken Teil.



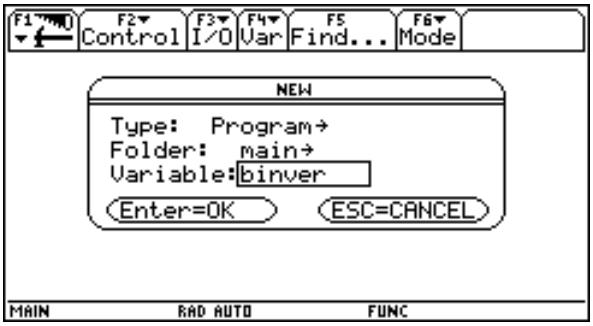

Dann holen wir über F5 den IO-Schirm dazu oder testen im Homebereich die Belegung von xc und yc.



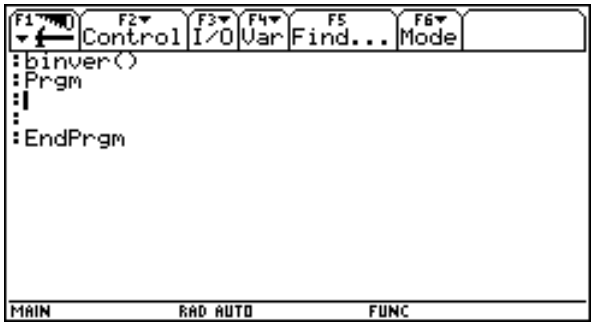



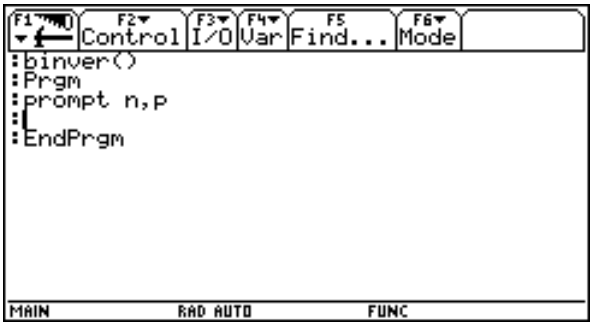
# BINOMIALVERTEILUNG

Nun wollen wir uns der Berechnung und Darstellung der Binomialverteilung zuwenden. Wir öffnen ein neues Programm mit dem Namen binver.

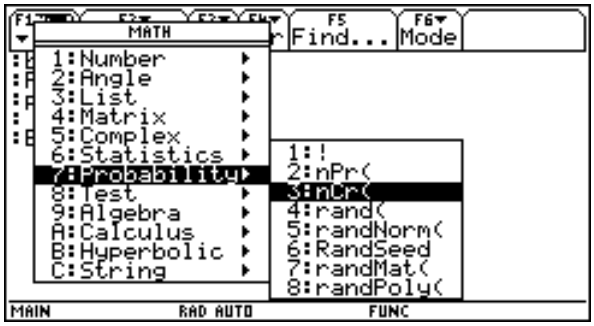



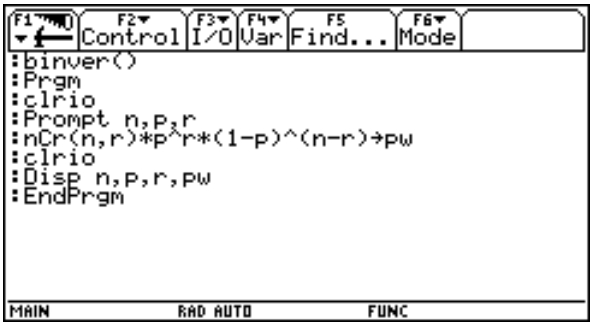

Es erscheint die übliche Programmstruktur. Über den Promptbefehl wollen wir die Anzahl n und die Wahrscheinlichkeit p eingeben.



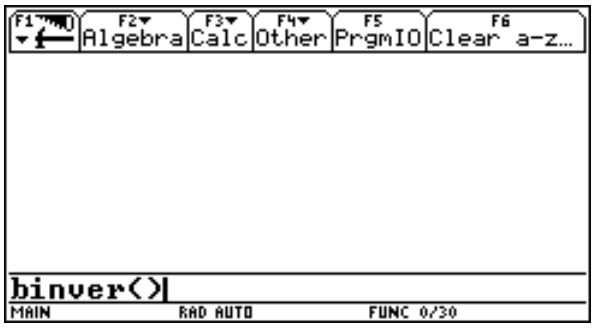



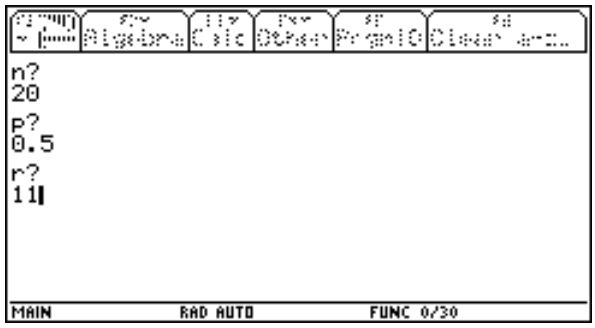

Zur Berechnung der Wahrscheinlichkeit P(X=r) benötigen wir den Befehl nCr, der uns den Binomialkoeffizienten berechnet., nCr(n,r) = .(nr). Wir finden den Befehl im Katalog oder im Menü MATH bei Probability. Außerdem löschen wir vor dem Promptbefehl mit ClrIO den Bildschirm, ergänzen den Promptbefehl um die Variable r und geben die Eingabegrößen und die berechnete Wahrscheinlichkeit mit disp aus, nachdem auch hier zuvor mit ClrIO der Bildschirm gelöscht wurde.



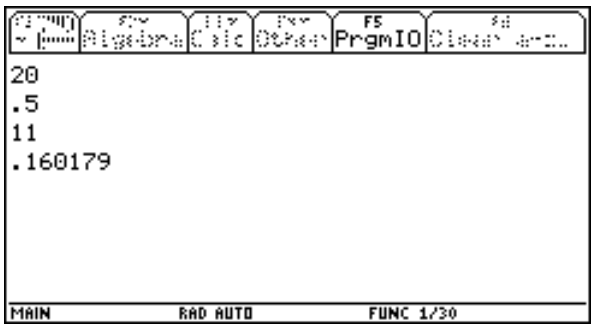



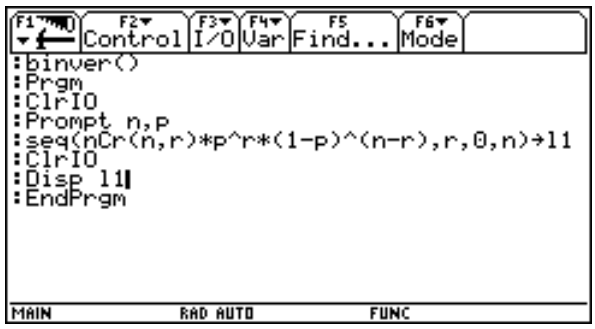
Wir kehren in den Homebereich zurück und testen das Programm.



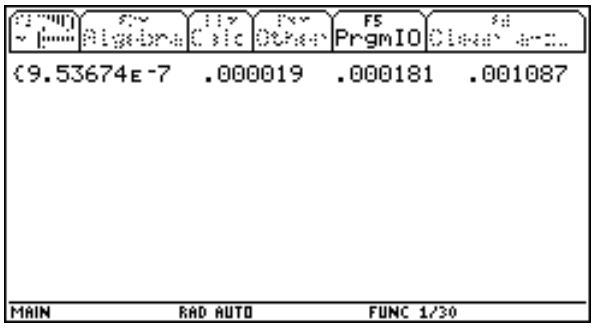



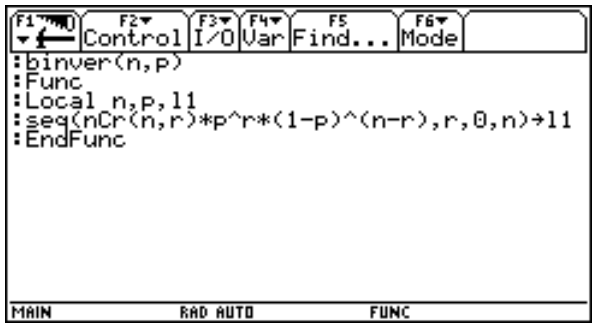

Wir erhalten folgende Ausgabe. Wir verändern nun das Programm, um die gesamte Verteilung auf einmal berechnen zu können. Der Befehl seq erzeugt uns eine Liste l1, beginnend mit der Wahrscheinlichkeit P(X=0) bis zur Wahrscheinlichkeit P(X=n).



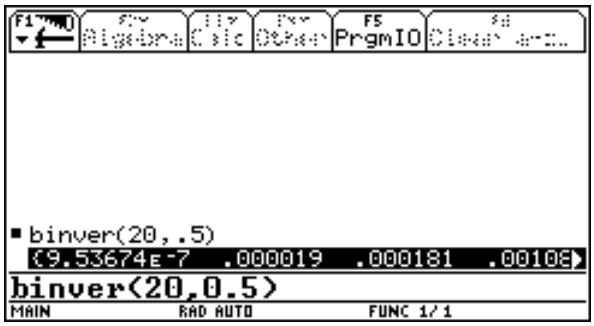



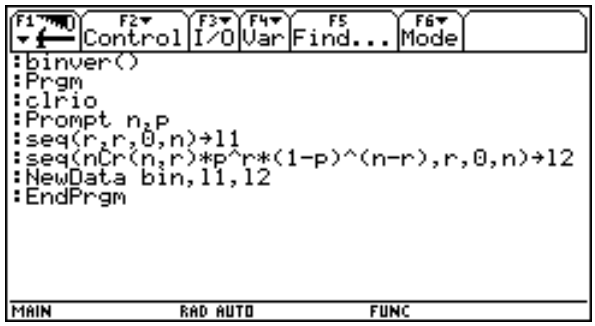

Die Ausgabe dieser Liste über disp am IO-Schirm erweist sich als ungünstig. Der Befehl disp kennt keinen Zeilenumbruch und ein Scrollen der Liste ist auch nicht möglich.
Ein Ausweg wäre eine Funktion zu programmieren und dabei n und p als Parameter zu übergeben.



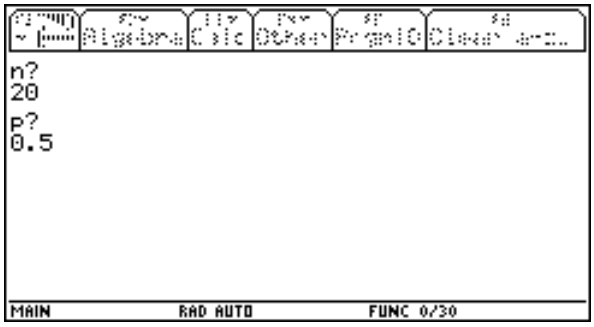




Die Ausgabe dieser Funktion erscheint nun im Homebereich und man kann in der Liste Scrollen. Trotzdem ist es nicht bequem für n=50 die Wahrscheinlichkeit P(X=39) zu suchen. Wir kehren daher zu unserem Programm zurück und wollen die Listen des Data/Matrixeditors verwenden. Dazu schaffen wir mit seq eine Liste l1, die von 0 bis n verläuft, und eine Liste l2 mit den entsprechenden Wahrscheinlichkeiten dazu. Mit dem Befehl NewData erzeugen wir ein neues Datenblatt bin, dessen 1. Spalte c1 aus der Liste l1 und dessen 2. Spalte c2 aus der Liste l2 besteht.



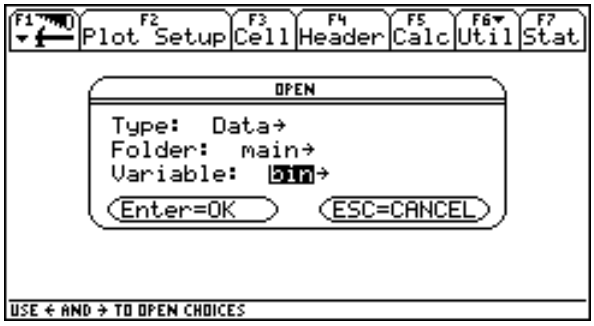



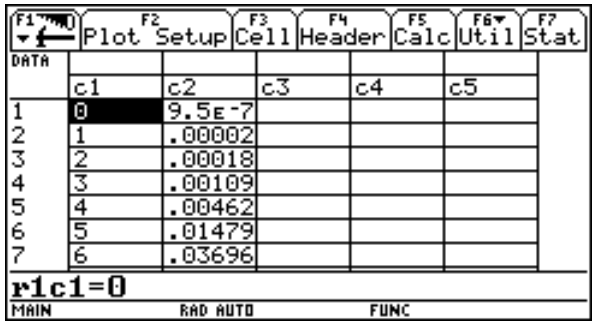

Wir testen das veränderte Programm. Nach Ablauf des Programmes öffnen wir im Data/Matrixeditor das Datenblatt bin.



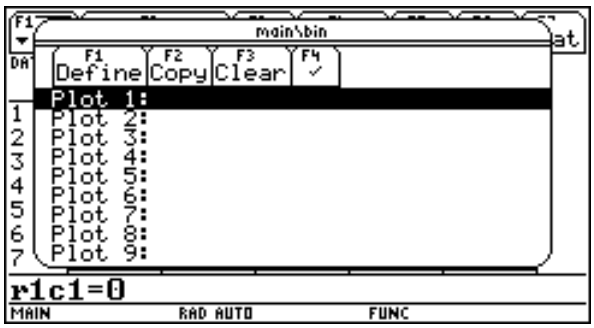


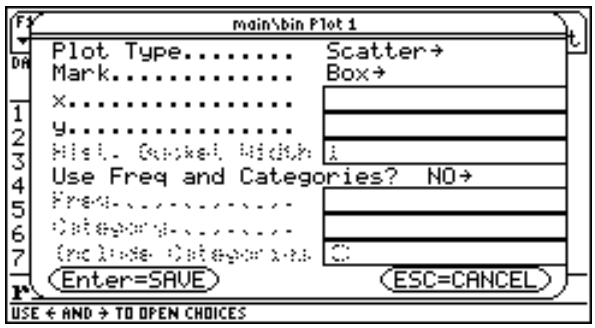
Da die Zeilen am linken Rand des Datenblattes mitgezählt werden, ist die Spalte c1 nur deshalb notwendig, weil wir die Verteilung auch darstellen lassen wollen.



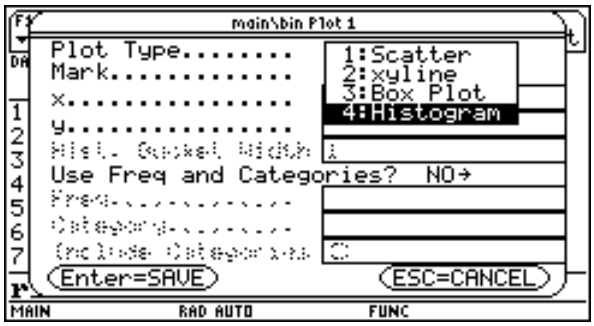



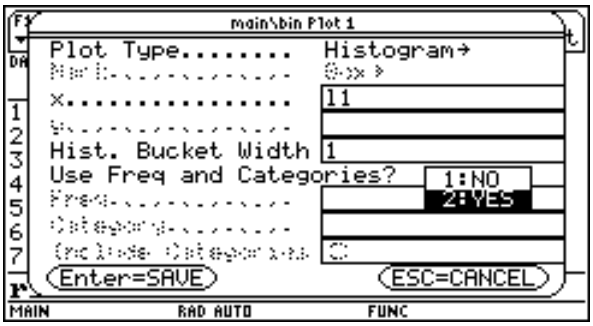

Dazu wählen wir F2/Plot Setup. Dann nach Auswahl von Plot 1 wählen wir F1/Define. Als erstes legen wir den Plottyp fest.



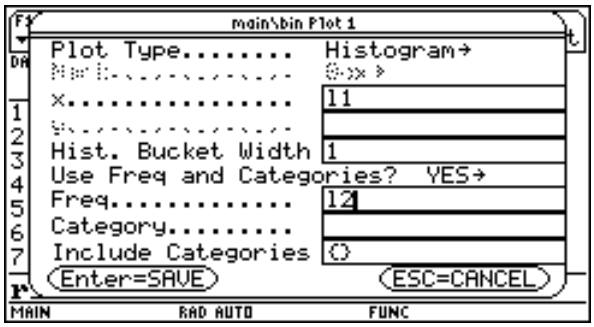



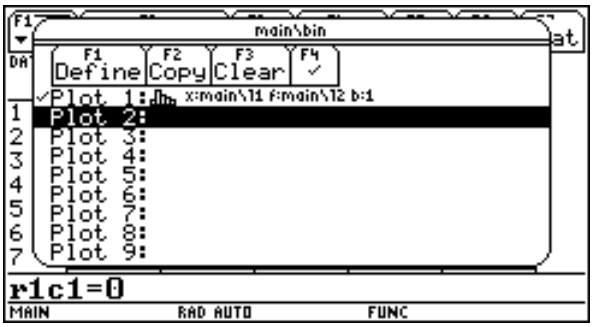
Wir entscheiden uns für Histogramm. Als Daten für die X-Achse verwenden wir die Liste l1, die Breite der Rechtecke des Histogramms belassen wir auf 1.
Die Häufigkeit der in der Liste l1 vorkommenden Werte wollen wir durch die Verwendung von Frequency festlegen. Daher wählen wir Yes.



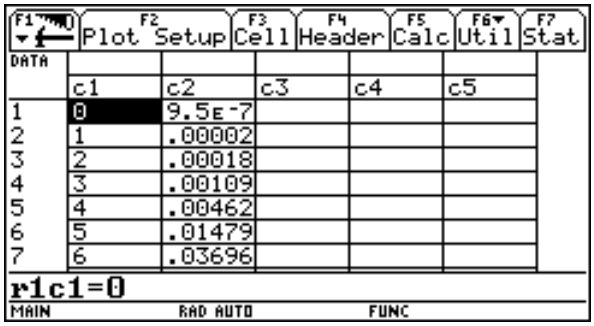



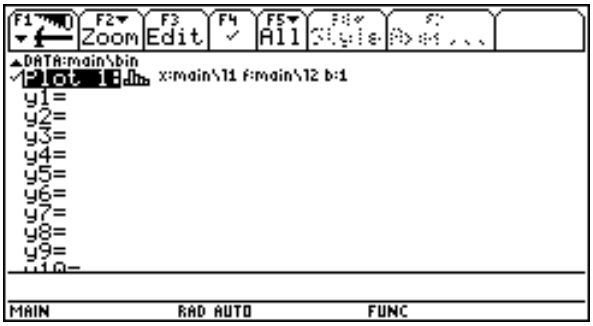

Und geben bei Frequency die Liste l2 mit den berechneten Wahrscheinlichkeiten ein. Ein Doppelenter führt uns einen Schritt zurück und zeigt uns in abgekürzter Schreibweise unsere Einstellung.



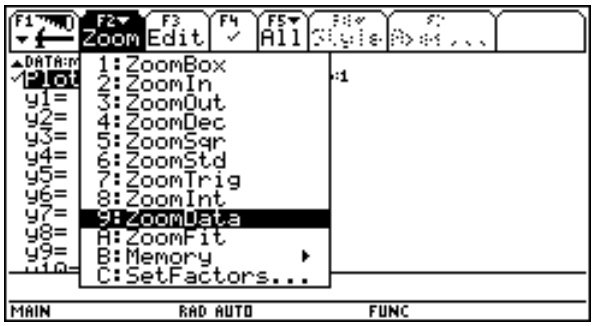



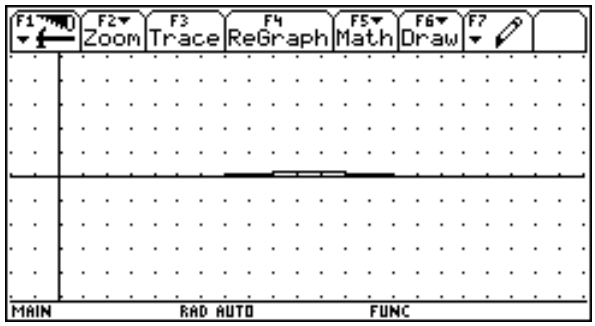

Noch ein Enter bringt uns ins Datenblatt zurück. Auch im Y-Editor können wir unsere Einstellung des Plots in abgekürzter Schreibweise kontrollieren.



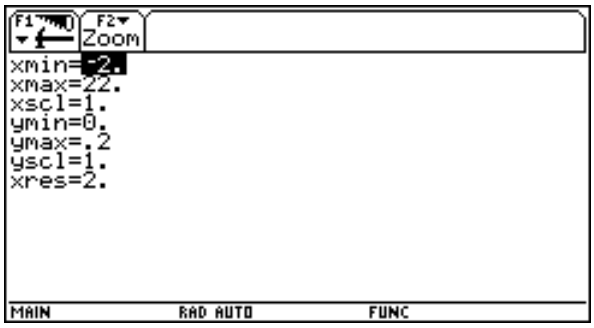



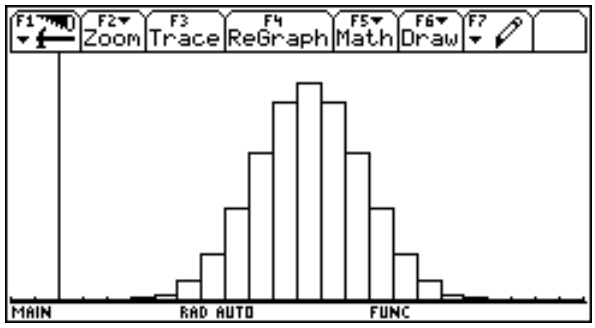

Mit ZoomData wird Xmin,Xmax an unsere Daten angepaßt. Ymin,Ymax müssen wir im Fall der Verwendung von Frequency noch selbst verändern.



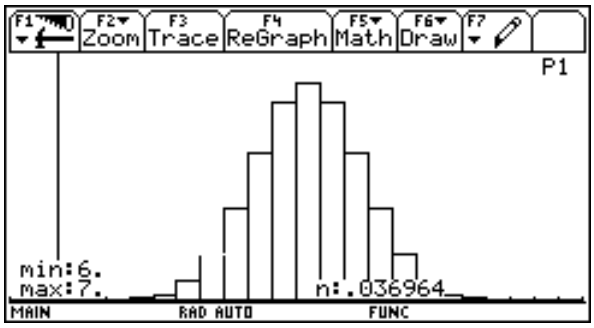



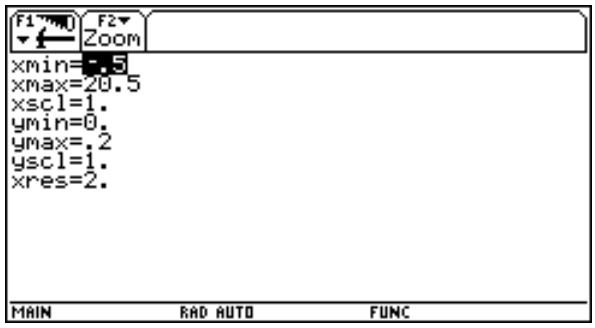

Mit den entsprechenden Windowvariablen erhalten wir folgende Darstellung.



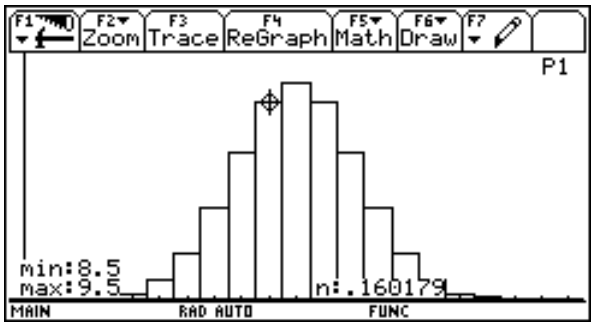



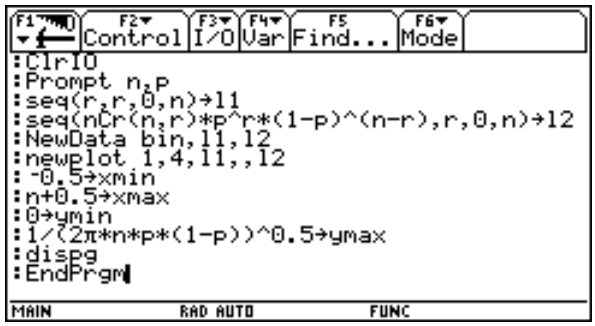

Mit F3 Trace bewegen wir uns entlang des Histogramms und bekommen die Breite und Höhe angezeigt. Wir wollen die Lage der Rechtecke noch so verändern, daß sie nicht von z.B. 6 - 7 sondern von z.B. 6,5 - 7,5 verlaufen. Das können wir dadurch verändern, daß wir xmin mit -0,5 festlegen.



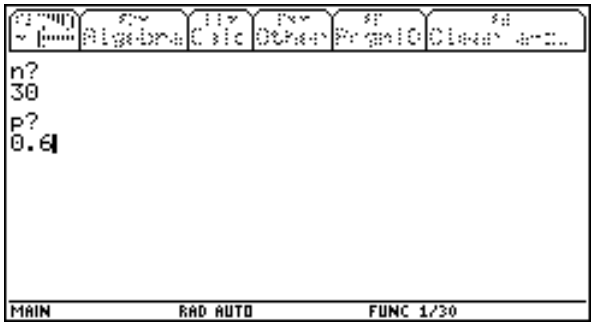



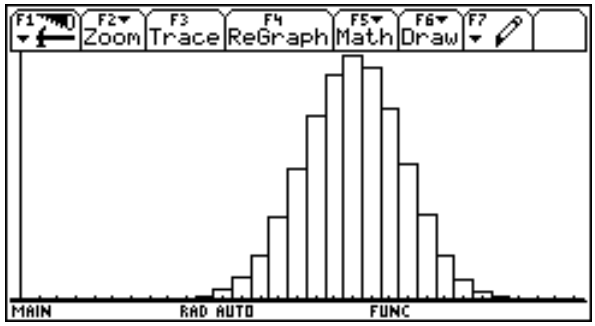

Dann ergibt sich die folgende Darstellung. Alle zuletzt getätigten Einstellungen wollen wir nun dem Programm übertragen. Dazu dient der Befehl NewPlot. An der 1. Position wird die Nummer des Plots festgelegt, bei uns Nummer 1, an der 2. Position wird die Art des Plots festgelegt, bei uns 4, was einem Histogramm entspricht, an der 3. Position die Liste für den x-Bereich, bei uns l1, an der 4. Position die Liste für den y-Bereich, bei uns leer und an der 5. Position die Liste für Frequency, bei uns l2. Dann legen wir die Windowvariablen fest, wobei wir Ymax durch das Maximum der zugeordneten Normalverteilung bestimmen.



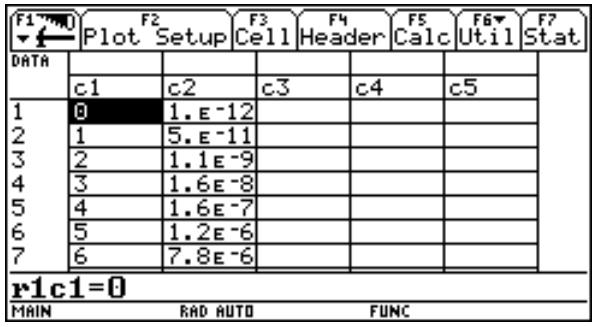




Ein Test unseres Programmes führt zu folgender Darstellung.

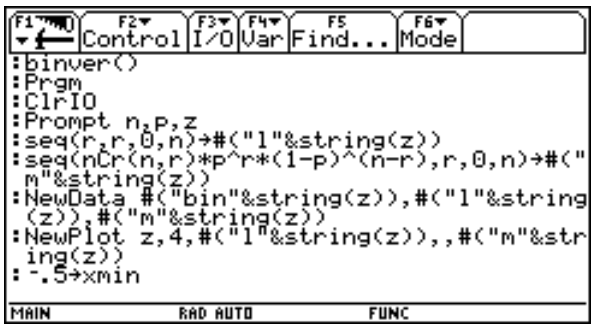


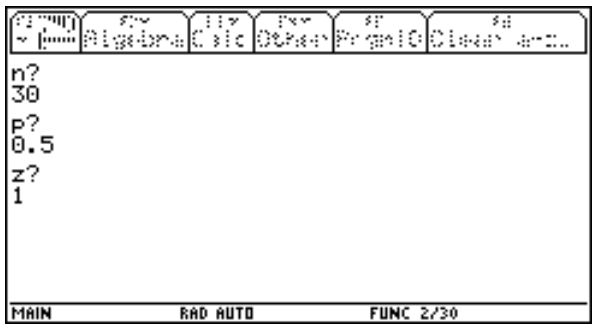




Im Data/Matrixeditor finden wir das entsprechnde Zahlenblatt.

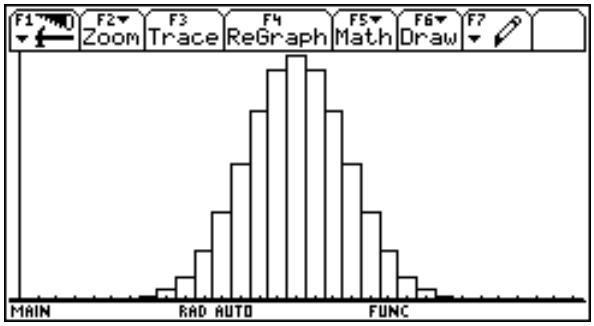



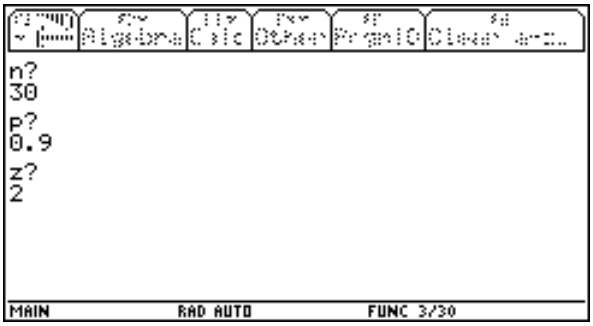

Nun wollen wir die Namen der beiden Listen und des Datenblattes und die Plotnummer von einer Variablen z abhängig programmieren, damit wir zum Vergleich auch mehrere Verteilung zugleich darstellen können. Wir ergänzen den Promptbefehl um die Variable z. Und indizieren mit z die beiden Listen und das Datenblatt. Zum Beispiel führt z=3 zu den Listen l3 und m3 und zum Datenblatt bin3 und der 3. Plot wird verwendet.
Diese Veränderungen müssen auch in den Befehlen NewData und NewPlot durchgeführt werden. Dann können wir wieder testen. Wir beginnen mit z=1.



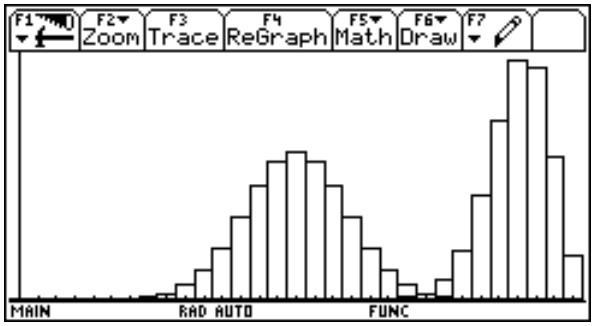



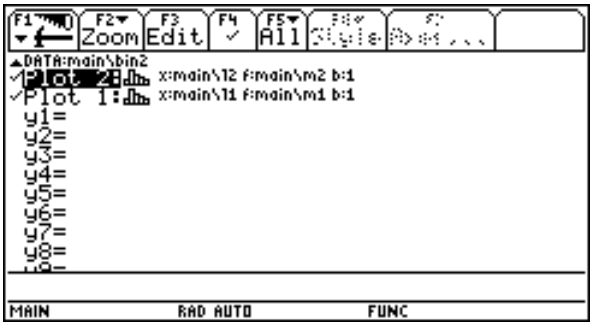
Erhalten folgende Darstellung und führen das Programm nocheinmal mit z=2 durch.



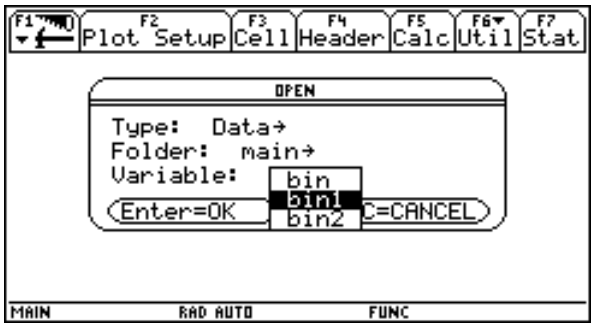



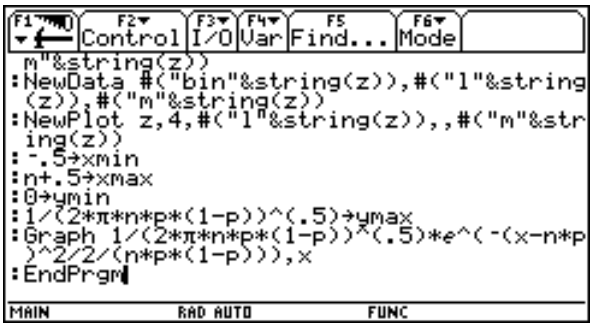

Beide Verteilungen erscheinen in der Graphik. Im Y-Editor sind die ersten beiden Plots belegt.



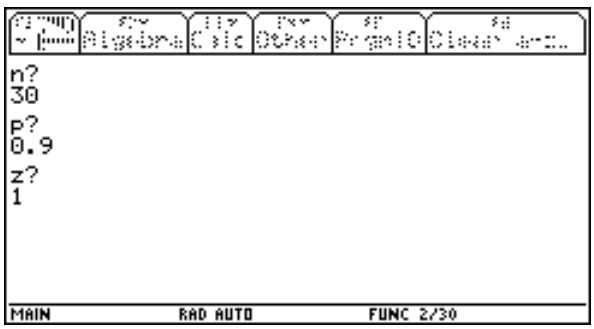



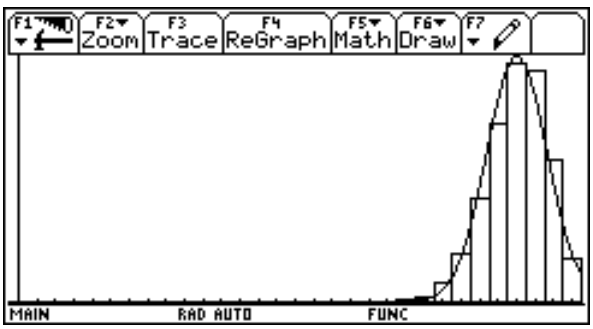

Zwei Datenblätter sind im Data/Matrixeditor angelegt. Mit dem Befehl Graph können wir nun noch erreichen, daß für die letzte Verteilung die zugeordnete Normalverteilung gezeichnet wird.




Es ergibt sich zum Beispiel folgende graphische Darstellung.

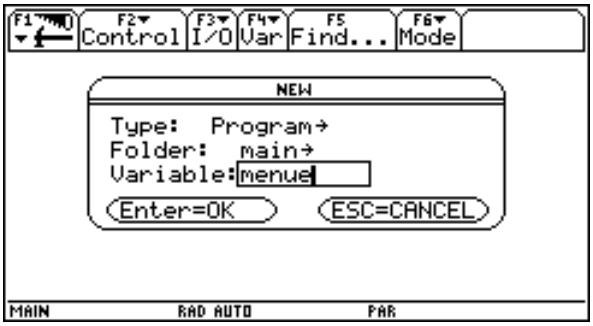


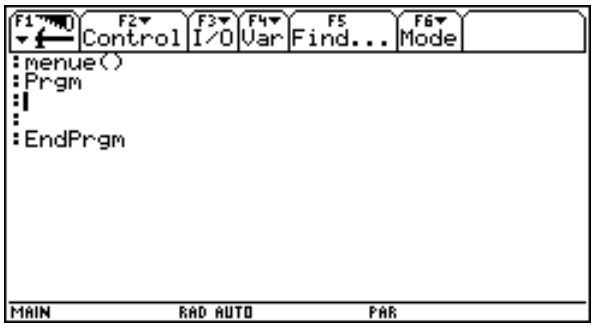
# BENUTZERDEFINIERTES MENÜ

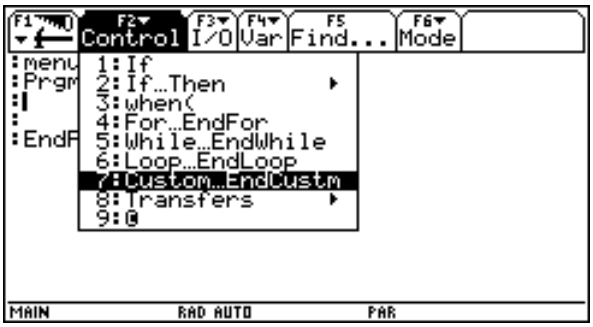
Es ist möglich zusätzlich zur Menüleiste des TI-92 eine eigene Menüleiste zu definieren. Über sie kann man eigene Programme und Funktionen bzw. Funktionen des TI-92, die man nicht immer über den Katalog oder im vierten Untermenü suchen möchte, aufrufen.

Wir starten die Programmierung eines neuen Programmes mit dem Namen “Menue”.

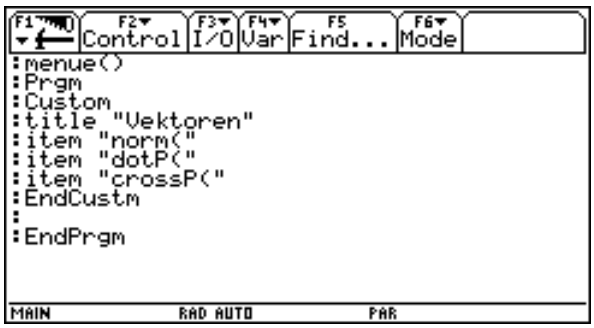



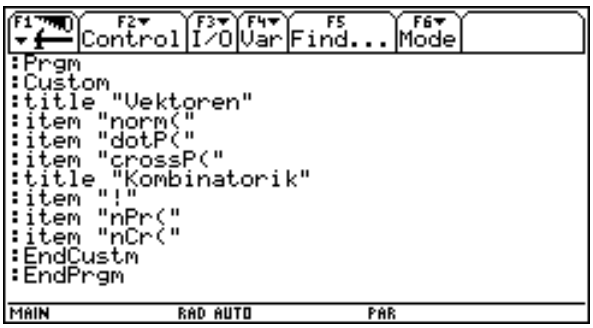

Es erscheint die übliche Programmstruktur, in die wir den Befehl Custom - Endcustm für den Aufruf eines benutzerdefinerten Menüs einfügen.



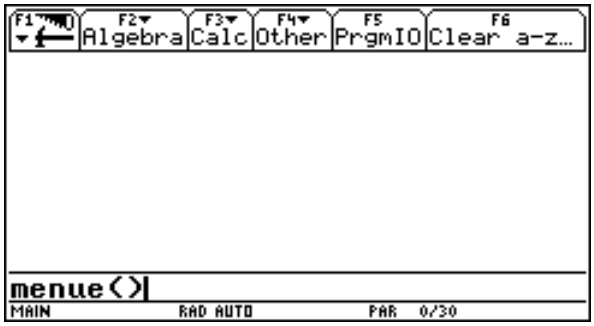



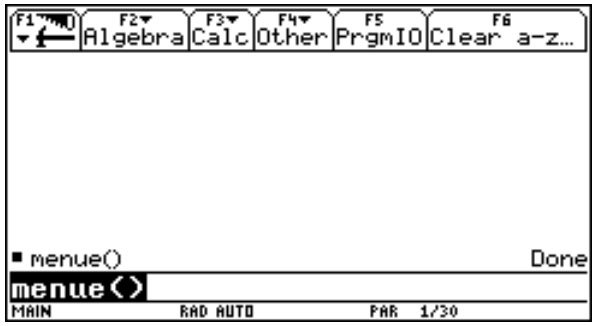

Mit dem Befehl Title wird die Überschrift des Pulldownmenüs, mit dem Befehl Item werden die einzelnen zu dieser Überschrift gehörenden Befehle festgelegt
Wir legen eine Menüleiste mit zwei Pulldownmenüs an, eines für Vektorrechnung und eines für Kombinatorik.



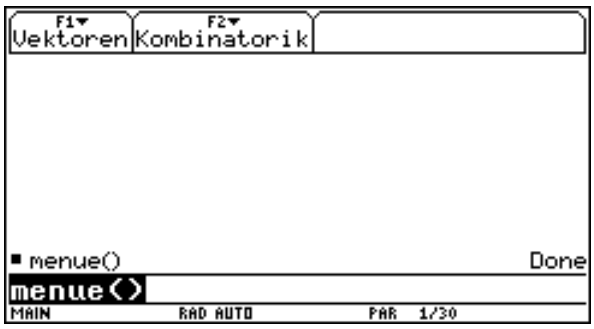



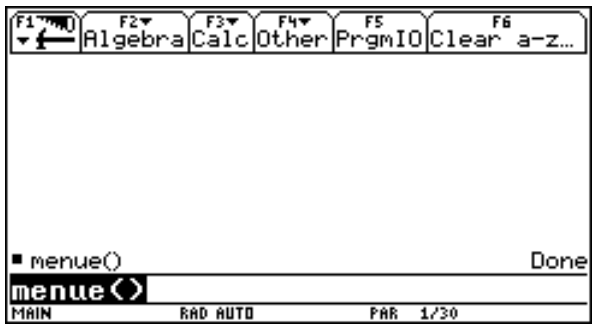
Wir testen das Programm. Es läuft ab und endet mit der Meldung “done”.



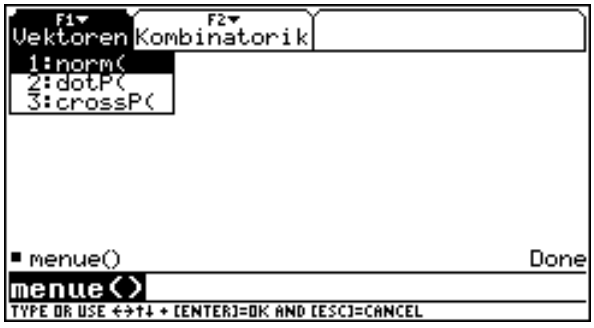



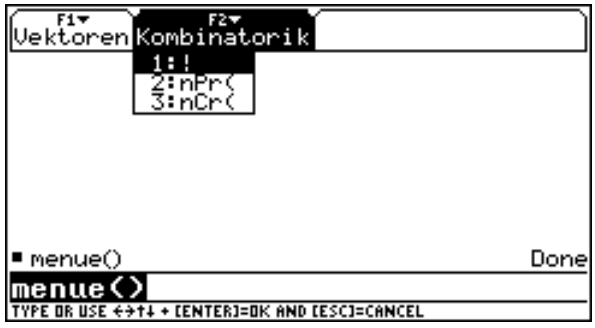

Über 2nd Custom können wir unsere Menüleiste aufrufen. Über 2nd Custom erhalten wir auch wieder die original Menüleiste. Man kann unter anderen Namen auch noch weiter Menüleisten anlegen. Aber neben der Originalmenüleiste kann immer nur eine vom Benutzer definerte Menüleiste aktiv sein.

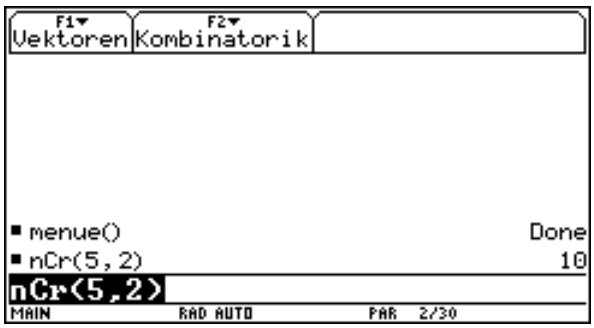



Die Menüleiste läßt sich wie gewohnt bedienen. Über Enter gelangt der gewünschte Befehl in die Eingabezeile.

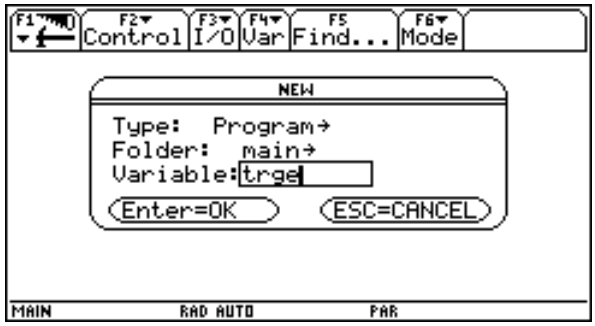


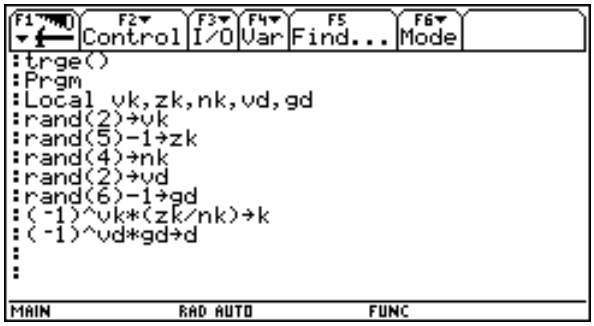
# ÜBUNGSPROGRAMM GERADENGLEICHUNG

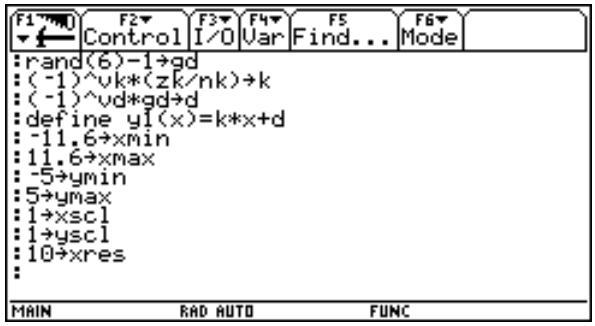
Dieses Programm soll über den Generator von Zufallszahlen eine lineare Funktion bestimmen, zeichnen, Anstieg k und Abschnitt d sollen vom Benutzer aus der Grahpik abgelesen werden, die Angaben des Benutzers über k und d sollen überprüft werden.
Wir beginnen eines neues Programm zu entwerfen und nennen es trge.

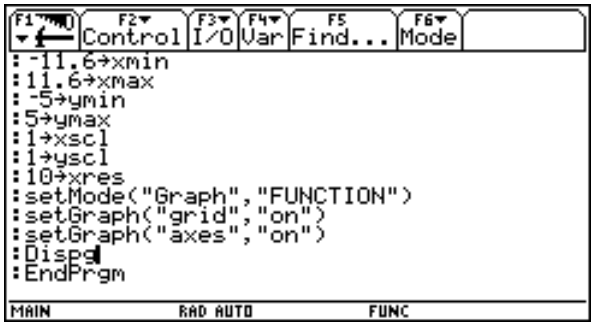

Zunächst wollen wir die zufällige lineare Funktion bilden. Der Befehl rand(2) weist der Variablen vk zufällig 1 oder 2 zu. Die Variable vk wird uns zur Berechnung des Vorzeichens vom Anstieg k dienen. Der Befehl rand(5)-1 weist der Variablen zk eine zufällige ganze Zahl zwischen 0 und 4 zu. Die Variable zk wird den Zähler des Anstieges bilden. Der Befehl rand(4) weist der Variablen nk eine zufällige ganze Zahl zwischen 1 und 4 zu. Die Variable nk wird den Nenner des Anstieges bilden. Der Befehl rand(2) weist der Variablen vd zufällig 1 oder 2 zu. Die Variable vd wird uns zur Berechnung des Vorzeichens des Abschnittes d dienen. Der Befehl rand(6)-1 weist der Variablen gd eine zufällige ganze Zahl zwischen 0 und 5 zu. Damit beschränken wir uns bei unseren zufällig bestimmten linearen Funktionen auf die Fälle, die aus einem Koordinatensystem leicht ablesbar sind.

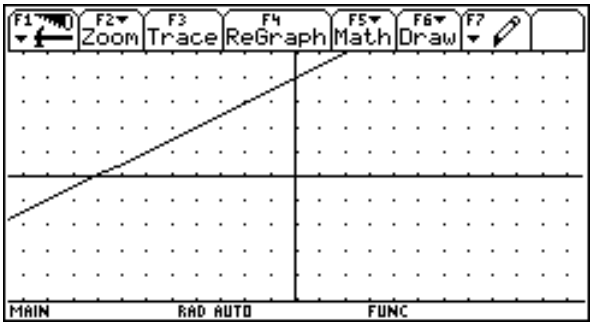
Alle oben genannten Variablen werden als local definiert. Die aus ihnen berechneten Variablen k und d bleiben global, weil sie zur Definition der Funktion y1 des Y-Editors verwendet werden. Die Windowvariablen werden nach dem Bereich für d und gleich großen Einheiten auf den Koordinatenachsen bestimmt (siehe Zoomsqr). xres wir auf 10 gestellt, um ein schnelles Zeichnen der Geraden zu ermöglichen.



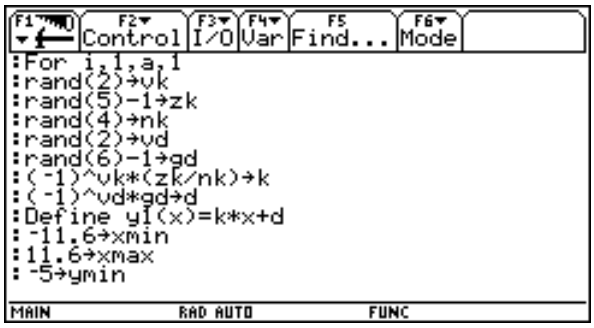



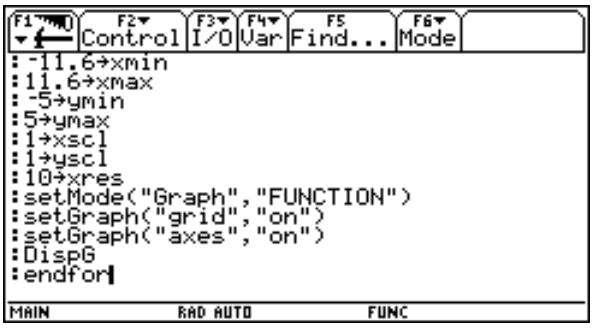
Der Graphikmode wird auf Function gestellt, Gitter und Koordinatenachsen werden eigeblendet. Über dispg wird der Graphikschirm aufgerufen. Ein erster Test des Programmes liefert folgenden Graphikschirm.



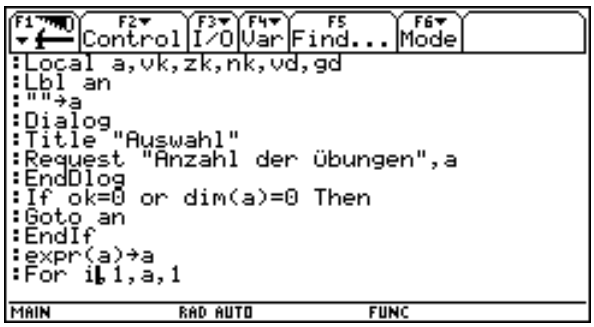



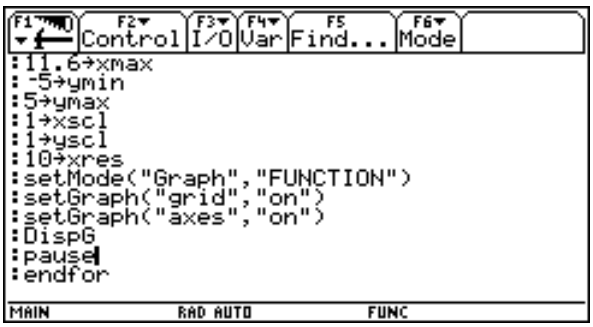

Nun bauen wir im Programm eine Schleife ein, die mehrer Übungsdurchgänge ermöglichen soll. Die Anzahl der Durchgänge, die Obergrenze der Schleife a, wollen wir späterhin vom Benutzer des Programmes festlegen lassen.



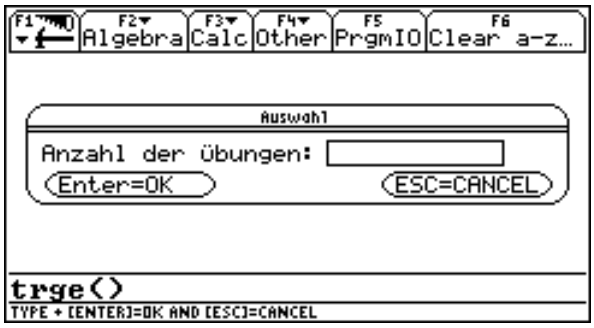



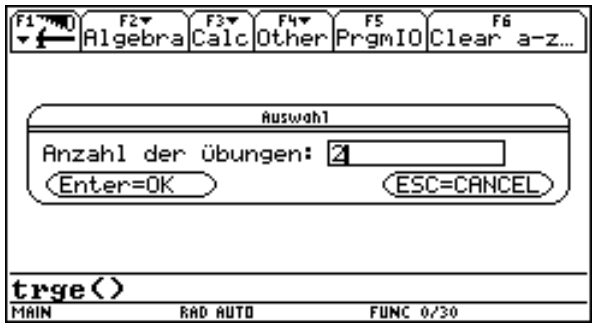

Dazu erstellen wir am Beginn des Programmes eine entsprechende Dialogbox. Die Abfrage nach der Dialogbox dient zur Kontrolle, ob die Box nicht mit ESC oder ohne Eingabe verlassen wurde. Damit dim(a)=0 nicht bei unbelegtem a zu einem Fehler führt, wird a vor der Box mit einem leeren String belegt. Nicht vergessen darf man auch die durch den Befehl Expr durchgeführte Verwandlung.



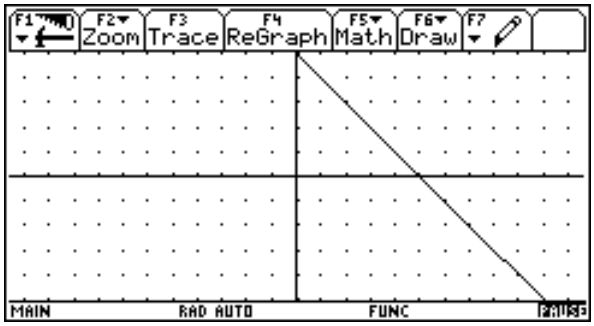



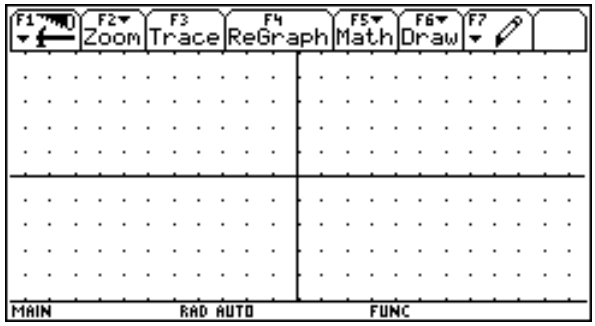
Ein Test liefert folgende Box.



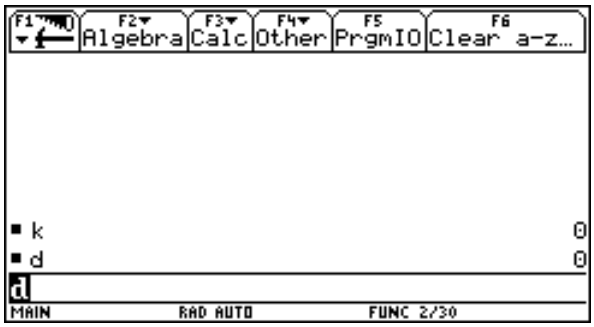



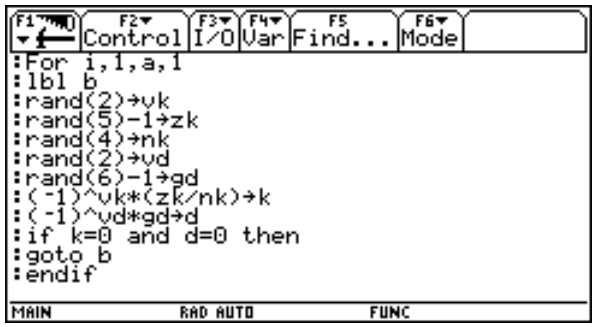

Und nach der Wahl von 2 z.B. die folgenden beiden Graphiken.




An der zweiten Graphik erkennen wir, daß es güngstig erscheint die Nullfunktion auszuschließen. Ein Blick in den Homebereich und der Aufruf der globalen Variablen k und d bestätigt diesen Verdacht. Mit einer Abfrage werden wir die Nullfunktion ausschließen.

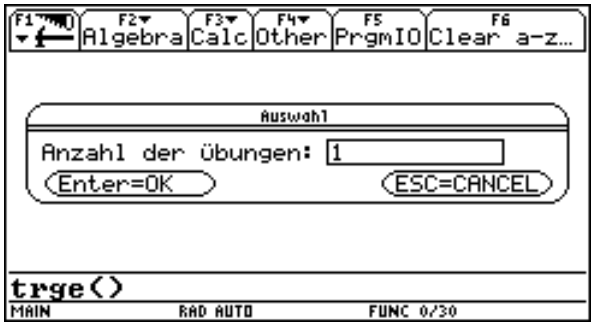


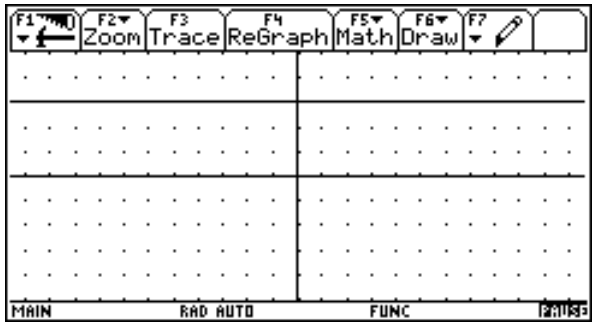




Nun legen wir die Dialogbox an, die den Benutzer auffordert, die aus der Graphik herausgelesenen Werte für k und d einzugeben.
Wir setzen zunächst einen Label b, um bei fehlender Eingabe oder Verlassen der Box mit ESC zum Aufruf der Box zurückkehren zu können. Dann werden den beiden Variablen k1 und d1 Texte ohne Buchstaben zugeordnet, damit die Abfragen dim(k1) bzw. dim(d1)=0 auch einen Sinn ergeben, wenn die Box ohne Eingabe verlassen wurde. Nach Ende des Aufrufes der Dialogbox folgt die Abfrage, ob die Box mit ESC ( ok=0 ) oder ohne Eingabe mit Enter ( dim(k1)=0 oder dim(d1)=0 ) verlassen wurde.
In diesen Fällen springen wir zum Label b zurück. Als Abschluß folgt noch die Verwandlung der unter k1 und d1 abgespeicherten Texte in Terme.

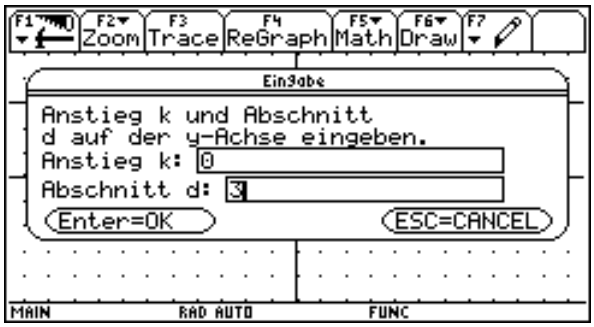


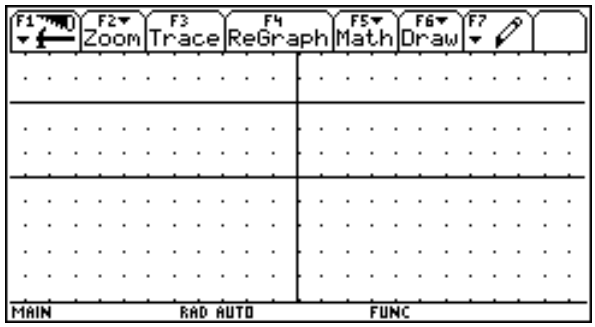
Testen wir nun unser Programm.



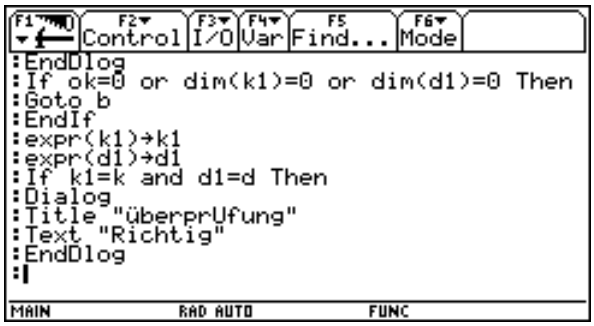




Auf die Eingabe der Anzahl der Durchgänge folgt der Graphikbildschirm und dann die eben erzeugte Dialogbox. Bei Verlassen mit ESC oder mit Enter ohne Eingaben gemacht zu haben, erscheint die Box neuerlich.



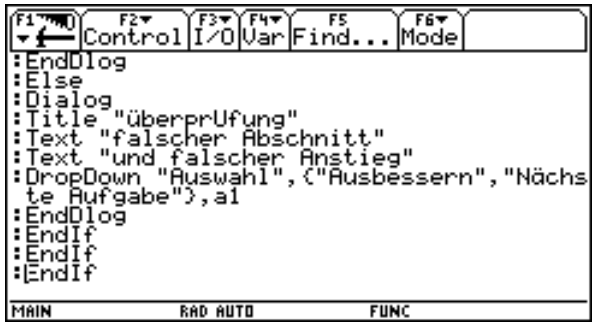




Nun wollen wir in Form von Dialogboxen die entsprechenden Anworten programmieren.
Dabei wollen wir durch drei Abfragen, die vier möglichen Fälle an Antworten durchprüfen: k und d richtig; k richtig, d falsch usw.
Zunächst wenden wir uns dem Fall zu, daß k und d richtig bestimmt wurden. Für diesen Fall enthält die Box nur einen einfachen Text.
Im Falle, daß k falsch und d richtig bestimmt wurden, enthält die Box zunächst einen Text, der auf den falschen Anstieg hinweist. Dann wollen wir die Möglichkeit bieten, den Fehler ausbessern zu können, also nocheinmal die Graphik zu sehen und neuerlich k und d eingeben zu können, oder zur nächsten Aufgabe zu schreiten. Dazu verwenden wir ein DropDown-Menü. Als Text geben wir “Auswahl” ein. Dann folgt eine Liste mit den zur Auswahl stehenden Texten. Die daran anschließende Variable erhält die Nummer des gewählten Listenelementes. Wird in unserem Fall “Anstieg ausbessern” gewählt, hat a1 die Belegung 1, wird “Nächste Aufgabe” gewählt, dann hat a1 die Belegung 2.

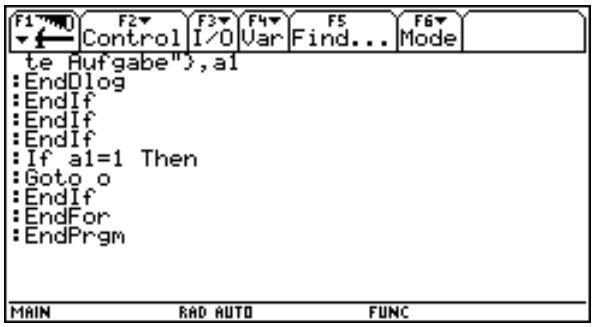


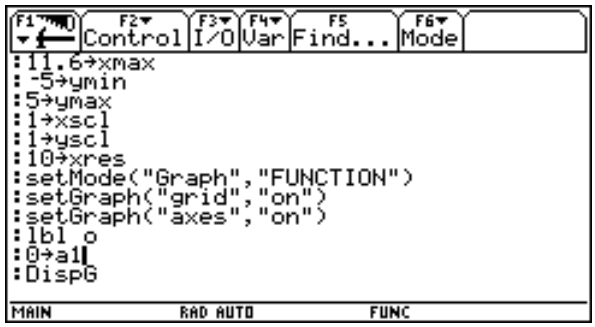


Analoge Boxen entwerfen wir für die anderen beiden Fälle.

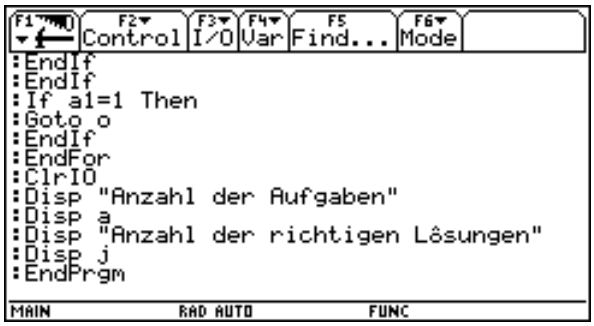



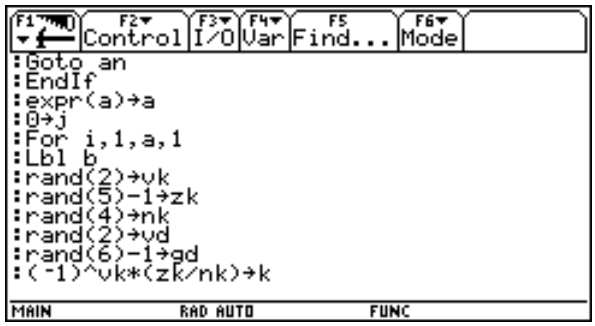

Dann prüfen wir durch eine Abfrage, ob a1=1 ist, also wiederholt werden soll oder nicht. Im Falle der Wiederholung springen wir zum Label O, den wir vor dem Aufruf der Graphik dispG setzten. Dort weisen wir der Variablen a1 den Wert 0 zu, damit die Abfrage a1=1 im Falle einer richtigen Anwort zur Fortführung des Programmes führt.



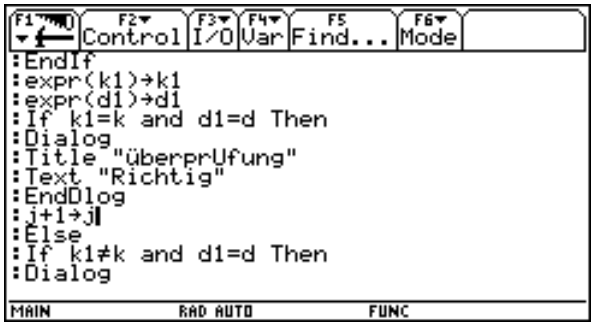



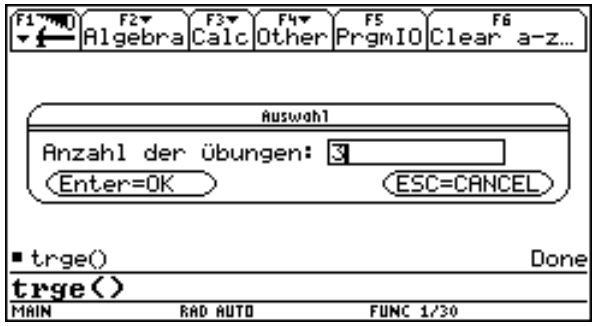

Am Ende unserer Schleife, alle Übungen sind vollendet, wollen wir noch ausgeben, wieviele Antworten richtig waren. Dies führen wir über disp-Befehle am IO-Schirm durch, nachdem wir diesen mit ClrIO gelöscht haben. Die Variable j sollte die Anzahl der richtigen Antworten enthalten. Wir setzen sie vor Beginn der Schleife gleich Null.



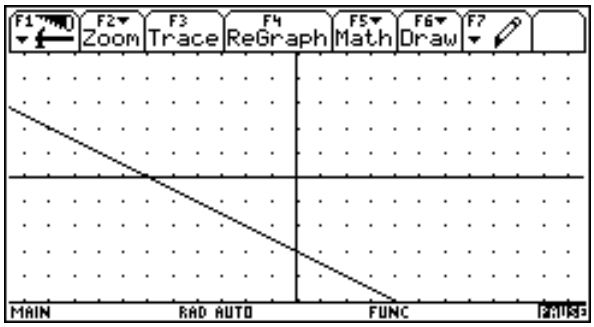



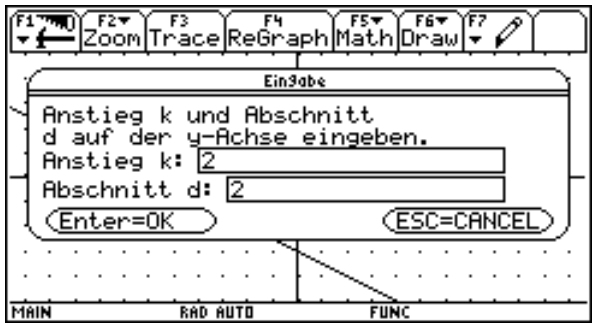

Und erhöhen sie bei jeder richtigen Antwort um eins.
Dann wollen wir das Programm ausprobieren. Wir wählen drei Übungsbeispiele.



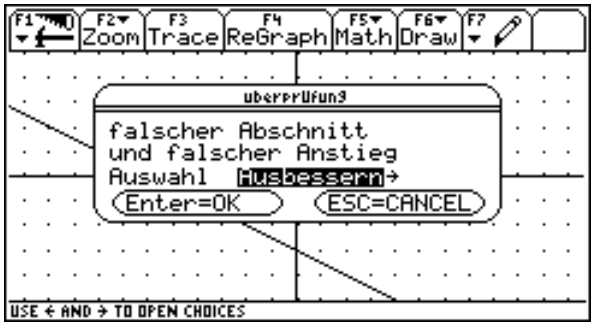



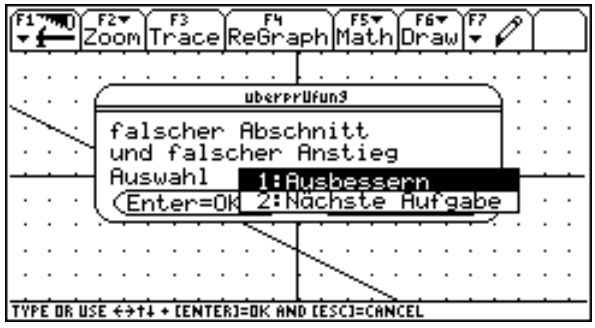

Dann geben wir k und d falsch ein.



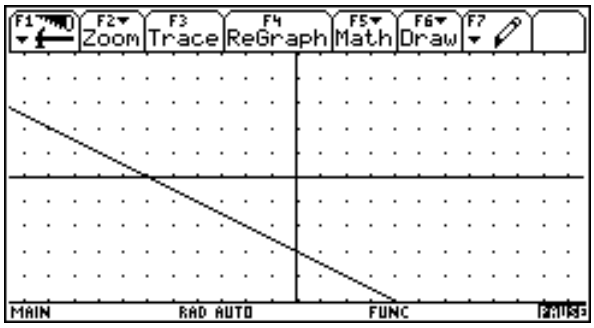


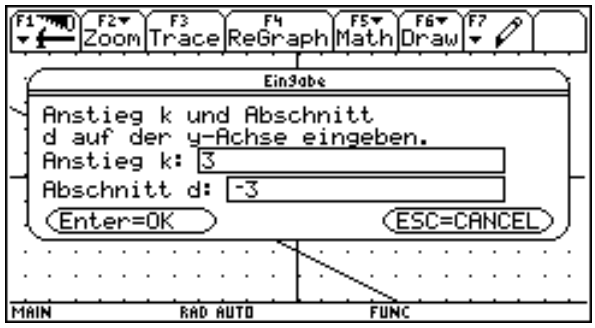

Wir erhalten die entsprechende Meldung und entscheiden uns für “Ausbessern”.



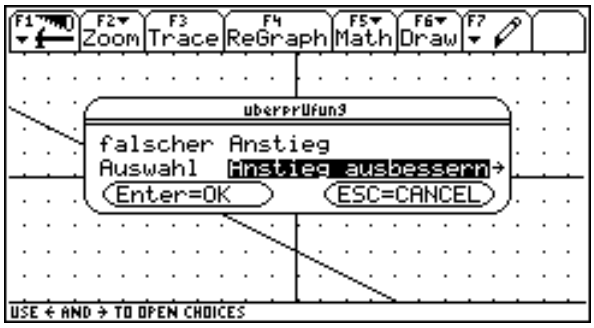



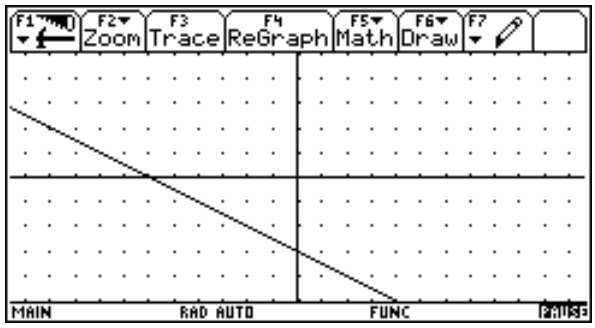
Das Programm kehrt zur Graphik zurück. Nun geben wir d richtig und k falsch ein.



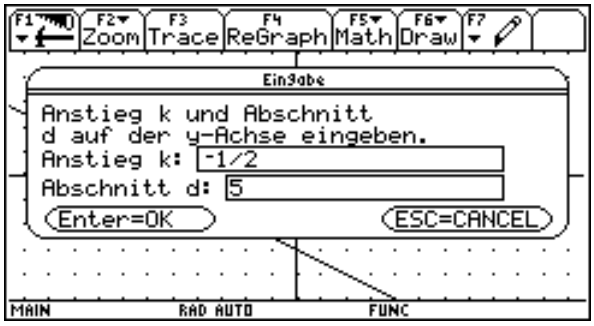



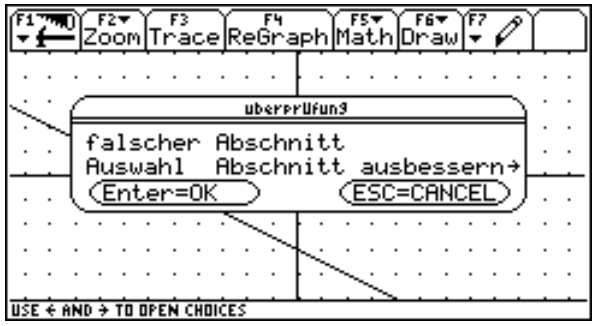

Wir erhalten die entsprechende Meldung und kehren über “Ausbessern” zur Graphik zurück.



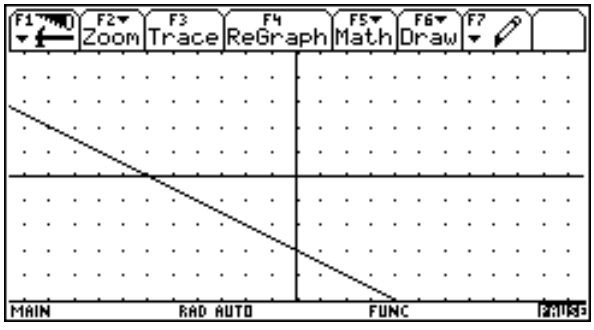



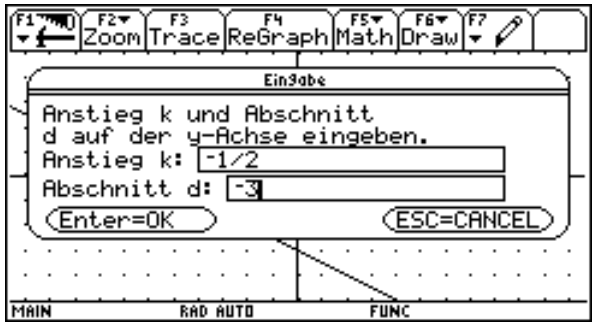

Nun geben wir den Anstieg richtig ein und den Abschnitt falsch. Wir erhalten wieder die entsprechende Meldung und kehren mit “Ausbessern” zur Graphik zurück.



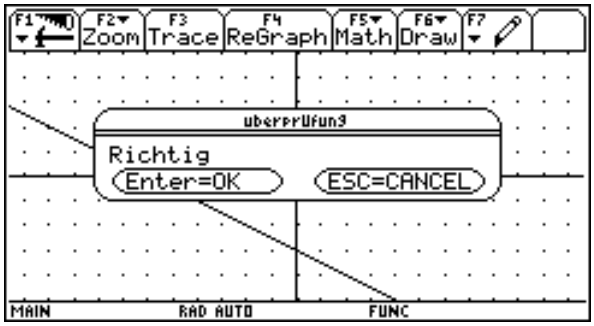



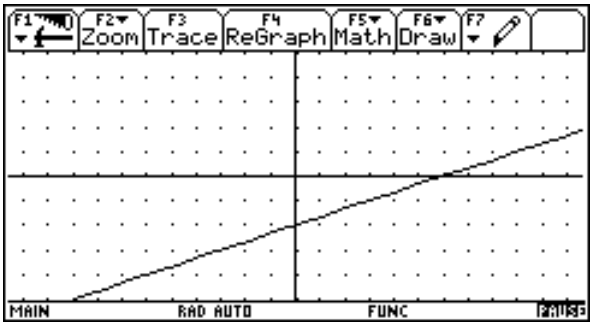

Zum Schluß geben wir k und d richtig ein.



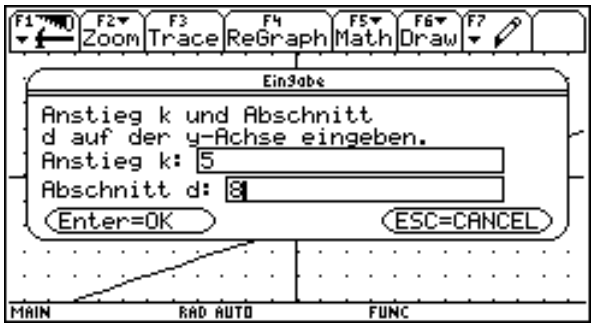



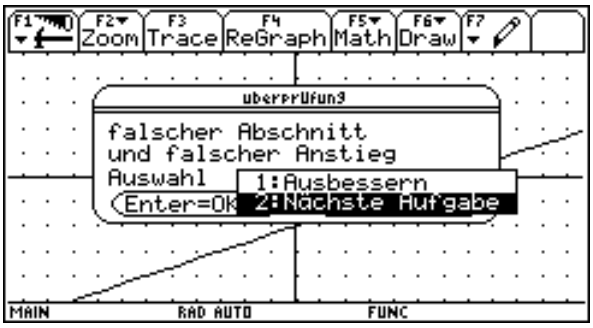
Es erscheint die Box für eine richtige Antwort.

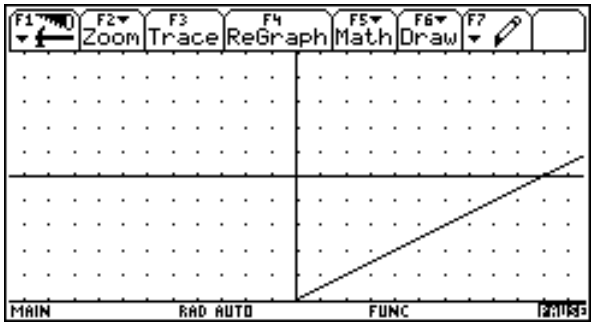


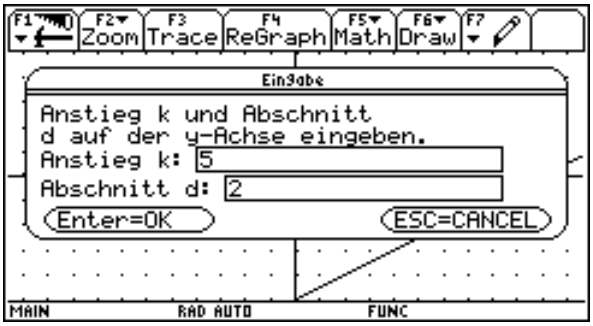




Die beiden nächsten Aufgaben übergehen wir, indem wir falsche Werte eingeben und “Nächste Aufgabe” wählen.

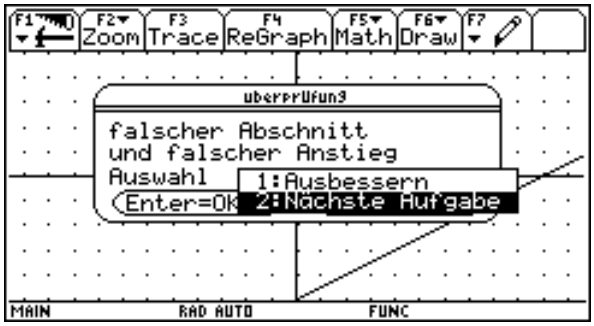


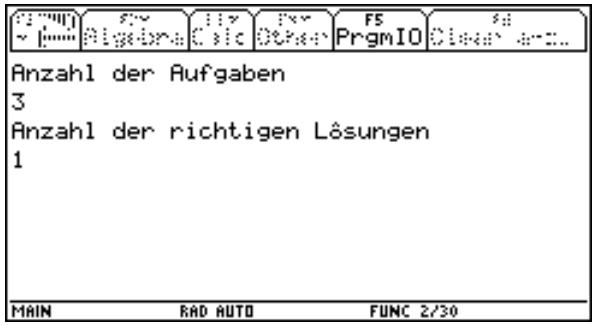



Am Ende erhalten wir die Statistik unserer Antworten.




Zum Schluß ergänzen wir noch die Liste der localen Variablen.