

# Programming an EL-9650/9900

Peter McIntyre

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Preliminaries</b>	<b>1</b>
<b>3</b>	<b>The basic program</b>	<b>2</b>
3.1	Writing the basic program . . . . .	2
3.2	Making corrections . . . . .	2
3.3	Making the program more user-friendly . . . . .	3
<b>4</b>	<b>Looping</b>	<b>3</b>
4.1	Looping for input . . . . .	3
4.2	Looping to check input . . . . .	4
4.3	Including an error message . . . . .	5
4.4	Looping through a range of input values . . . . .	6
<b>5</b>	<b>Displaying results as a table</b>	<b>7</b>
<b>6</b>	<b>Displaying results as a graph</b>	<b>8</b>
6.1	Graphing data points from a program . . . . .	8
6.2	Graphing functions from a program . . . . .	9
6.3	Putting text on a graph . . . . .	10
<b>7</b>	<b>Simple animation</b>	<b>12</b>
<b>8</b>	<b>Extensions</b>	<b>13</b>
<b>9</b>	<b>Exercises</b>	<b>16</b>
<b>10</b>	<b>Possible solutions to the exercises</b>	<b>17</b>
<b>11</b>	<b>Keystrokes used in the programs</b>	<b>20</b>

## Contact for more help

Peter McIntyre

School of Physical, Environmental  
and Mathematical Sciences  
University College (UNSW)  
Australian Defence Force Academy  
Canberra ACT 2600

Email: [p.mcintyre@adfa.edu.au](mailto:p.mcintyre@adfa.edu.au)  
Phone: (02) 6268 8896  
FAX: (02) 6268 8786

At [www.unsw.adfa.edu.au/pems/news/high\\_school/hsc\\_activities.html](http://www.unsw.adfa.edu.au/pems/news/high_school/hsc_activities.html)

- A variety of graphics-calculator activities for Years 9 and 10 — written as part of the CQTP Program for Sharp, Casio and TI calculators.
- *Using the EL-9650/9900* — an introduction to the basic operations, suitable for Years 8–12.
- *The Graphics Screen and Accuracy* — information to help you understand the graphical and numerical limitations of a graphics calculator.
- *Coordinate Geometry on an EL-9650/9900* — basic commands and a variety of problems, suitable for Years 9 and 10.
- *Population Modelling* — a variety of problems from simple exponential growth to Leslie matrices and difference equations, covering Years 7–12.
- *Sequences and Series on an EL-9650/9900* — basic commands and a variety of problems, suitable for Years 10–12.
- *Matrices on an EL-9650/9900* — suitable for Years 11 and 12.
- *Calculus on an EL-9650/9900* — suitable for Years 11 and 12.
- *Complex Numbers on an EL-9650/9900* — suitable for Years 11 and 12.
- *Introduction to Complex Numbers* — complex numbers from the beginning, covering the basic operations, but set in the context of complex numbers as a mathematical structure.

## Linking with a computer

The EL-9650/9900 can be linked to a computer using a special cable (which has to be bought) and software which comes with the cable.

You can use the software to copy programs from the calculator to the computer for storage or printing, and to ‘grab’ a screen image, which you can then print or store as a file to put in notes (very useful).

## 1 Introduction

A basic program structure is **INPUT** → **CALCULATION** → **OUTPUT**. The input is usually numbers (it might also be a function), the output can be text (e.g. ‘Complex Roots’), numbers, a graph or a table.

The programming language of the EL-9650/9900 is a subset of the BASIC programming language, containing most of the useful features of that language together with graphics commands.

Most of the commands have to be selected from a menu — you cannot type them in letter by letter. The PRGM menu selected from the home screen allows you to run, edit and create programs, while PRGM selected while editing a program gives access to most of the programming commands.

The hardest thing to start with is finding the appropriate commands on the keyboard and in menus.<sup>1</sup> *On pages 20 and 21 of these notes is a list of key strokes for the commands used in the programs here.* The Table of Functions at the back of the EL-9650/9900 Guidebook is also very useful.

## 2 Preliminaries

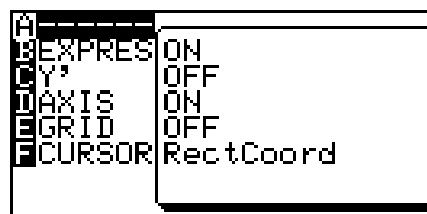
In what follows, we refer to the key with +, −, × and ÷ on it (first column, second row) as the *Home* key. Pressing this returns you to the Home screen.

Before starting, press **SETUP** (**[2ndF]** **[BS]**) and set up your calculator as shown in the figure.



<b>A</b>	-----
<b>B</b> URG	Rad
<b>C</b> FSE	FloatPt
<b>D</b> TAB	9
<b>E</b> COORD	Rect
<b>F</b> ANSWER	Decimal(Real)
<b>G</b> EDITOR	One line
<b>H</b> SIMPLE	Auto

Then press **FORMAT** (**[2ndF]** **[ZOOM]**) and set up your calculator as shown in the figure.



<b>A</b>	-----
<b>B</b> EXPRES	ON
<b>C</b> Y'	OFF
<b>D</b> AXIS	ON
<b>E</b> GRID	OFF
<b>F</b> CURSOR	RectCoord

---

<sup>1</sup>If you're really stumped, all the commands are in the CATALOG menu (**[2ndF]** **[÷]**): press **[ALPHA]** and then a letter key to go to commands starting with that letter. Scroll down to the bottom to get to the commands or characters that don't contain letters.

## 3 The basic program

We start with a simple example and successively add features.

*Input the radius of a circle, calculate and output its area.*

### 3.1 Writing the basic program

Press the **PRGM** key and move to NEW with the cursor. Press **ENTER**.

Enter the name of the program (up to 8 characters) — CIRCLE. Note that the calculator is in ALPHA mode. Press **ENTER** and you are now ready to enter the program.

Press **PRGM** and have a look at the commands in each of the sub-menus.

Here is our basic program. Press **ENTER** at the end of each line, including the last line. Press **CL** in a menu to return to the program.

Input R    **PRGM** (PRGM) **3** **ALPHA** **R**

$\pi R^2 \Rightarrow A$     **2ndF** **(-)** **ALPHA** **R** **x<sup>2</sup>**  
                  **STO** **ALPHA** **A**

Print A    **PRGM** (PRGM) **1** **ALPHA** **A**

<b>CIRCLE</b>
Input R
$\pi R^2 \Rightarrow A$
Print A

Don't forget to press **ENTER** after the last line.

When you are happy with the program, press the Home key to return to the main screen.

Press **PRGM**, select EXEC and then the CIRCLE program by typing its number or with the cursor and **ENTER**.

Try some R values (R = 1 should give a recognisable area) to make sure the program is running correctly.

Note that if you press **ENTER** after the program has finished, you will rerun the program. This makes it easy to enter successive R values.

Press the Home key when you have finished running the program.

### 3.2 Making corrections

If you have made a mistake in entering the commands, the calculator may detect the error and stop with an error message and a choice of quitting (returning to the home screen) or going to the error. Select the second option by pressing the left or right arrow and the calculator will take you to the problem in the program.

If the program runs OK, but gives the wrong output, press **PRGM**, select EDIT and then the program. This allows you to edit the program.

In editing a program, note the **DEL** and **INS** keys. Press **INS** again to turn INS off. **CL** clears a whole line of text. **DEL** on a blank line deletes the line.

Press **ENTER** or the up or down arrow after correcting a line to store the changes.

### 3.3 Making the program more user-friendly

It makes the program easier to use if the program gives some idea of what it does, what input it requires and what it outputs.

Commands that are new or changed from the previous program are highlighted.

`INS` `ENTER` at the beginning or end of a line gives a blank line.

`2nd` `ALPHA` gives ALPHA LOCK (for letters); `ALPHA` turns it off.

The symbol `□` indicates a space (`ALPHA` `□`). You only need to put in a space when you see this symbol.

Press `PRGM` EDIT and select the CIRCLE program.

```
Print "AREA□OF□A□CIRCLE"  PRGM (PRGM) 1 PRGM 2 2ndF ALPHA
                             A ... E ALPHA PRGM 2
```

You need to turn off A-LOCK before the final `PRGM` `2` ("): hence the `ALPHA`.

Input `"RADIUS"`, R

$\pi R^2 \Rightarrow A$

Print `"AREA"`

Print A

Press the Home key, then `PRGM` EXEC and run this version.

## 4 Looping

### 4.1 Looping for input

Maybe we'd like to enter a number of values for R without having to rerun the program each time. We do this with a simple loop using a *Label* and a *Goto*.

Print "AREA OF A CIRCLE"

Label 1      `PRGM` BRNCH

Input "RADIUS", R

$\pi R^2 \Rightarrow A$

Print "AREA"

Print A

Goto 1      `PRGM` BRNCH      Press `ENTER` to store this line.

This makes entering a number of radius values easier, but how do we stop the program?

Press `ON` `CL`. This stops any program and returns to the home screen. Use `ON` and the left- or right-arrow key to go back to edit the program — useful during the development stages.

## 4.2 Looping to check input

Another place for a loop is to test whether an input value falls within a specified range. In our example, we expect  $R \geq 0$  (although it doesn't matter in this case because we square  $R$ ). Let's use a loop to test for this. We also introduce the *If* statement here.

Print "AREA OF A CIRCLE"

Label 1

Input "RADIUS", R

If  $R < 0$  Goto 1      *If* is in **PRGM** BRNCH; < is **MATH** INEQ **5**.<sup>2</sup>

*Do this step if the 'If' is true, otherwise skip this step*

$\pi R^2 \Rightarrow A$

Print "AREA "

Print A

Goto 1

Check your program with suitable inputs, i.e. at least one negative  $R$ .

See Extension 1 on page 13 for another way to use such a loop and Extension 2 on page 13 for a more elegant error check.

---

<sup>2</sup>The calculator displays this line as *If R<0Go to 1*. The spaces in the notes are for clarity — you should not insert a space.

### 4.3 Including an error message

Let's say we want to tell the user that he or she has made a mistake in the input. Such diagnostics are always useful, especially for students.

Here we embellish the *If* statement with a *Then* and *End*,<sup>3</sup> and introduce *Wait*, which stops the program until the user presses `ENTER`.<sup>4</sup>

That a program is 'Waiting' is indicated by a flashing rectangle in the top right corner of the screen.

Note that anything in inverted commas is text, and can be whatever you want.

```

Print "AREA OF A CIRCLE"

Label 1

Input "RADIUS", R

If R < 0      delete the Goto 1 in the previous version
Then        PRGM BRNCH
Print "====_MAKE R ≥ 0_===="   = is ALPHA (-)   used for emphasis here
Print "PRESS ENTER"   PRGM PRGM
Wait        PRGM PRGM 4
Goto 1
EndIf      PRGM BRNCH   ends the If Then

 $\pi R^2 \Rightarrow A$ 

Print "AREA"

Print A

Goto 1

```

Don't forget suitable tests for this version of the program.

<sup>3</sup>necessary except when there is a *Goto* with the *If*.

<sup>4</sup>*Wait n* causes the calculator to pause for *n* seconds. For example, *Wait 5* could be used here instead of just *Wait*.

## 4.4 Looping through a range of input values

Here we use a *For* loop to carry out the calculations for a specified range of (equally spaced) input values. The *For* command has the format

*For*  $R, F, L, S$

*commands*

*Next*

$R$  starts with a value of  $F$  and executes *commands*.  $R$  is then increased<sup>5</sup> by  $S$  and *commands* are executed again. This continues until  $R$  is greater than or equal to  $L$ .

Remove the lines (use `CL` `DEL`) that test for  $R < 0$  in the previous program to keep the program simple.

Print "AREA OF A CIRCLE"

Input "FIRST\_R", F      *prompt for the initial value of R*

Input "LAST\_R", L      *prompt for the final value of R*

Input "R\_STEP", S      *prompt for the increment in R*

For R, F, L, S      `PRGM` BRNCH (scroll down)

$\pi R^2 \Rightarrow A$

Print "R, AREA"

Print R

Print A

Wait      *pause to read answer*

Next      `PRGM` BRNCH      *end For loop*

Run the program with suitable inputs, say  $R$  from 0 to 10 in steps of 1. Press `ENTER` to see successive answers.

---

<sup>5</sup> $S$  can be negative, so that  $R$  is decreased. In this case,  $L$  must be less than  $F$ , and the program stops when  $R \leq L$ .

## 5 Displaying results as a table

Rather than displaying the areas one by one as we did in the previous example, we can put them in a table. In a table, we can scroll up and down to see all the answers.

To make up a table, we need to put the R and A values in lists. Here we use list L1 for R and list L2 for A.

Lists L1 – L6 are built in to the calculator and can be accessed from the keyboard.

The variable I in the following program is the list index — we have to increment it by 1 each time we store a value in a list.

The LIST menu is `2ndF` `STATPLOT`.  $\Rightarrow$  is the `STO` key.

```
1  $\Rightarrow$  dim(L1)      dim( is LIST OPE 3; L1 is 2ndF 1  initialise list L1
```

```
1  $\Rightarrow$  dim(L2)      initialise list L2
```

```
1  $\Rightarrow$  I           initialise the list index
```

```
Print "AREA OF A CIRCLE"
```

```
Input "FIRST R", F
```

```
Input "LAST R", L
```

```
Input "R STEP", S
```

```
For R, F, L, S
```

```
R  $\Rightarrow$  L1(I)
```

```
 $\pi R^2 \Rightarrow$  L2(I)
```

```
I + 1  $\Rightarrow$  I      increment the list index by 1
```

```
Next
```

```
Print ""          blank line in the output
```

```
Print "PRESS STAT EDIT"  tell the user how to view the answers
```

Run the program with R varying from 0 to 10 in steps of 1 and look at the results. Press the Home key to return to the home screen from the table.

## 6 Displaying results as a graph

### 6.1 Graphing data points from a program

Once we have the results in lists, it is a simple step to graph them. We use `STATPLOT` to define the plot, i.e. tell the calculator where the data are, what sort of graph to plot and what marker to use for the points.

We also use *Zoom Stat* to set suitable axes for the graph and `TRACE` to allow us to move the cursor along the plotted values. We keep the table of values.

`DRAW` is `2ndF` `WINDOW`; `VARS` is `2ndF` `X/θ/T/n`.

`DrawOFF`      `DRAW` ON/OFF `2`    *turn off any functions in* `Y=`

`PlotOFF`      `PRGM` S\_PLOT `5`    *turn off any existing plots*

`ClrG`          `PRGM` SCRn `2`    *clear the graphics screen*

`0 ⇒ Xscl`      Xscl is `VARS` WINDOW XY `3`    *no tick marks on the X axis*

`0 ⇒ Yscl`      Yscl is `VARS` WINDOW XY `6`    *no tick marks on the Y axis*

`1 ⇒ dim(L1)`

`1 ⇒ dim(L2)`

`1 ⇒ I`

Print "AREA OF A CIRCLE"

Input "FIRST R", F

Input "LAST R", L

Input "R STEP", S

For R, F, L, S

R ⇒ L1 (I)

$\pi R^2$  ⇒ L2 (I)

I + 1 ⇒ I

Next

Plt1 (Scattr `□`, L1, L2)    `PRGM` S\_PLOT `1`    `STATPLOT` S.D. `3`

`Zm_Stat`      `ZOOM` ZOOM `9`

`TRACE`        TRACE *key*

Print ""

Print "PRESS STAT EDIT"

Run the program to test it. Pressing the left/right arrow keys will move the cursor along the plotted points. Press `ENTER` to exit from the graph.

## 6.2 Graphing functions from a program

Graphing a function from a program is simple — the function has to be defined in  $\boxed{Y=}$ , either before the program is run or from within the program. Be careful in a program that the right function is turned on — use DrawON and DrawOFF to do this. If you define a function within a program, it is automatically turned on.

There are several commands you can use in a program to display a graph. All of these will cause any functions *and any plots* turned on to be graphed.

- TRACE — displays the graph with a cursor you can move along the functions/points plotted, just as you would using TRACE manually. TRACE pauses the program — press  $\boxed{\text{ENTER}}$  to continue.
- DispG —  $\boxed{\text{PRGM}}$  SCRN  $\boxed{4}$ . *DispG* displays the graph without a cursor. *DispG* does not pause the program — you have to follow it with a *Wait*.
- Draw —  $\boxed{\text{DRAW}}$  DRAW  $\boxed{6}$ . *Draw* followed by a function definition either explicitly in terms of X (e.g.  $3X + 4$ ) or one of Y1 – Y0, draws that function. Use *Wait* to view the graph. A function drawn with *Draw* cannot be TRACED. An advantage of *Draw* is that it allows successive graphs to be superimposed — see Exercise 1.

To continue our example and do some curve fitting, let's fit a quadratic function to our data points, store the fitted function in Y1 and graph it together with the data. We expect a good fit! The data-fitting commands are in the  $\boxed{\text{STAT}}$  REG menu.

```

DrawOFF
PlotOFF
ClrG
0  $\Rightarrow$  Xscl
0  $\Rightarrow$  Yscl
1  $\Rightarrow$  dim (L1)
1  $\Rightarrow$  dim (L2)
1  $\Rightarrow$  I
Print "AREA OF A CIRCLE"
Input "FIRST R", F
Input "LAST R", L
Input "R STEP", S
For R, F, L, S
R  $\Rightarrow$  L1 (I)

```

```
 $\pi R^2 \Rightarrow L2(I)$ 
```

```
I + 1  $\Rightarrow$  I
```

```
Next
```

```
Plt1(Scattr□, L1, L2)
```

```
Zm_Stat
```

```
TRACE
```

```
Rg_ $x^2$ (L1, L2, Y1)  [STAT] REG; Y1 is [VARS] EQVARS [ENTER] [1]
```

```
[TRACE]      view the data points and the fitted curve
```

```
Print ""
```

```
Print "PRESS STAT EDIT"
```

Run this program and see if you get the expected fit to the data points. Press `[ENTER]` after the points are plotted to see the fitted curve. Then try the up and down arrows as well as the left and right arrows. Press `[ENTER]` to finish the program. You can then look at Y1 in `[Y=]` or at the data in `[STAT] EDIT`.

### 6.3 Putting text on a graph

Sometimes it is useful to annotate graphs so that the user can understand what is being plotted or be prompted for the next key stroke(s).

As an example, in the graph of our last two versions of the program, the calculator pauses in TRACE. Possible keys to press are the arrow keys to move along or between the graphs and `[ENTER]` to finish the program. We use the *Text* command to indicate this on the graph.

The *Text* command is of the form *Text*(*column*, *row*, "*text*"). The coordinates *column* and *row* are screen coordinates rather than the coordinates given by the WINDOW: *column* varies from 0 (left of screen) to 30 (right of screen), while *row* varies from 0 (top of screen) to 9 (bottom of screen). It is usually necessary to experiment with the coordinates to get the text in the right place.

The program is over the page.

```

DrawOFF
PlotOFF
ClrG
0 ⇒ Xscl
0 ⇒ Yscl
1 ⇒ dim (L1)
1 ⇒ dim (L2)
1 ⇒ I
Print "AREA OF A CIRCLE"
Input "FIRST R", F
Input "LAST R", L
Input "R STEP", S
For R, F, L, S
R ⇒ L1 (I)
 $\pi R^2$  ⇒ L2 (I)
I + 1 ⇒ I
Next
Plt1 (Scattr □, L1, L2)
Zm_Stat
Text(15, 0, "[ARROWS]_L[ENTER]")
[ DRAW ] DRAW [ 0 ]
[ is [ MATRIX ] [ ] [ 1 ] ]
TRACE
Rg- $x^2$  (L1, L2, Y1)
Text(10, 1, "[ARROWS]_L[ENTER]")
TRACE
Print ""
Print "PRESS STAT EDIT"

```

## 7 Simple animation

Animation is achieved by creating the separate frames, storing them in the picture memories Pict1, Pict2, ..., Pict0 and then recalling them one by one. Usually you need a suitable delay, created here with *Wait 1*, while each frame is displayed.

### Program ANIMATE

```
DrawOFF
PlotOFF
AxisOFF      [PRGM] FORMAT (scroll down)
Zm_Default   [ZOOM] [5]  standard axes
Zm_Square    [ZOOM] [6]  same scale on each axis
ClrG         clear the graphics screen
Shade( $-\sqrt{64 - X^2}$ ,  $\sqrt{64 - X^2}$ ) [DRAW] DRAW (scroll down)
                                     shade between the two functions
StoPict 1    [DRAW] PICT menu  store the first screen in Pict1
ClrG         clear the graphics screen
Circle(0,0,8) [DRAW] DRAW menu  circle centre (0,0), radius 8
StoPict 2    store the second screen in Pict2
Label 1     start the animation loop
ClrG         clear the graphics screen
RclPict 1   [DRAW] STO menu  recall the first screen
Wait 1     pause to display the first figure
ClrG         clear the graphics screen
RclPict 2   recall the second screen
Wait 1
Goto 1     go back to Label 1 (start of loop)
```

To stop the program, press [ON] [CL].

Clearly quite sophisticated animations can be programmed, given time. You might like to explore SLIDESHOW, a built-in animation program to make the job easier.

When you have finished, turn the axes back on in [FORMAT] AXIS and delete Pict1 and Pict2 using [OPTION] DEL.

## 8 Extensions

### Extension 1

Follows on from the example on page 4.

We could also use  $R < 0$  as a way to stop the program, rather than using `ON` `CL`. Note the message telling the user this after the main heading.

```
Print "AREA OF A CIRCLE"  
Print "R<0┘TO┘STOP"  
Label 1  
Input "RADIUS", R  
If R < 0  
Then    PRGM BRNCH  
End    PRGM PRGM  stop the program  
EndIf  
 $\pi R^2 \Rightarrow A$   
Print "AREA"  
Print A  
Goto 1
```

### Extension 2

Follows on from the example on page 4.

For a simpler, more elegant check on the input, we can use a *While* loop. The *While* loop makes sure that the input is positive or zero before the calculations are carried out.

```
Print "AREA OF A CIRCLE"  
Print "R<0┘TO┘STOP"  
Label 1  
-1  $\Rightarrow R$     make sure starting value of R is negative  
While R < 0  PRGM BRNCH (scroll down)  
Input "RADIUS", R  
WEnd    PRGM BRNCH
```

$\pi R^2 \Rightarrow A$

Print "AREA"

Print A

Goto 1

### Extension 3

Here we set up a custom menu in a program that calculates the area of a triangle, rectangle or circle. Only an outline of the program is given here — the details are to be completed in Exercise 2.

Note the use of a suggestive letter for each label.

#### Program AREA

Label M

Print "AREA OF A... "

Print " "

Print "1: TRIANGLE"

Print "2: RECTANGLE"

Print "3: CIRCLE"

Print "4: QUIT"

Key A     PRGM PRGM 7     waits for a numerical input 0 – 9

If A=1 Goto T

If A=2 Goto R

If A=3 Goto C

If A=4 Goto Q

Goto M     *return to start if input is not correct*

Label T

*Insert prompts, inputs and calculations to calculate area of a triangle.  
Store the area in A.*

Goto D

Label R

*Insert prompts, inputs and calculations to calculate area of a rectangle.  
Store the area in A.*

Goto D

Label C

*Insert prompts, inputs and calculations to calculate area of a circle.  
Store the area in A.*

Label D

Print "AREA "

Print A

Wait

Goto M

Label Q

## 9 Exercises

If these exercises seem too hard, think up a suitable calculation and program it.

### Exercise 1

Assume that  $Y1 = AX^2 + B$  and that a suitable WINDOW has been set. Your program called FAMILY should

- turn off all functions
- turn off stat plots
- clear the graphics screen
- display a header on the home screen
- start a loop (*Label*)
- input values for A and B
- draw the graph (Draw Y1 DRAW DRAW)
- pause to allow you to look at the graph
- return for another set of values for A and B (end of loop)

This program allows you to see the effect on the function of changing the parameters A and B.

Use a WINDOW of  $[-5, 5, 1] \times [-10, 10, 2]$ . Try  $A = 1, -1, 2, 0.5$  with  $B = 0$ . Use ON CL to stop the program and rerun with  $A = 1$  and  $B = 0, \pm 2, \pm 5$ .

You can put in Y1 any function containing two parameters A and B, and plot the corresponding family of curves using FAMILY. Try, for example,  $Y1 = A(X-B)^2$ .

### Exercise 2

Write a user-friendly program that calculates the area of a triangle, rectangle or circle, given appropriate inputs. *Hint*: see Extension 3.

### Exercise 3

Find the real roots of the quadratic equation  $Ax^2 + Bx + C = 0$ , where  $A$ ,  $B$  and  $C$  are inputs. Display a message if the roots are complex (later you might like to extend the program to complex roots). Finally plot the quadratic: specify the  $x$  range in the program ( $Xmin$ ,  $Xmax$ ) — the roots should help here — and use *Zoom Auto*, which calculates a suitable  $y$  range and plots the graph.

To put the quadratic function in Y1: " $AX^2 + BX + C$ "  $\Rightarrow$  Y1. Recall that Y1 is in VARS EQVARS.

## 10 Possible solutions to the exercises

### Exercise 1

#### Program FAMILY

DrawOFF

PlotOFF

ClrG

Print "FAMILY\_PROGRAM"

Print "PARAMETERS\_A,B"

Label 1

Input A

Input B

Draw Y1

Wait

Goto 1

## Exercise 2

### Program AREA

```
Label M
ClrT   [PRGM]SCRN   clears the text screen
Print "AREA_OF_A... "
Print "1: TRIANGLE"
Print "2: RECTANGLE"
Print "3: CIRCLE"
Print "4: QUIT"
Key A
If A=1 Goto T
If A=2 Goto R
If A=3 Goto C
If A=4 Goto Q
Goto M

Label T
ClrT
Print "      TRIANGLE"
Input "BASE", B
Input "HEIGHT", H
0.5BH  $\Rightarrow$  A
Goto D

Label R
ClrT
Print "      RECTANGLE"
Input "LENGTH", L
Input "WIDTH", W
LW  $\Rightarrow$  A
Goto D

Label C
ClrT
Print "      CIRCLE"
Input "RADIUS", R
 $\pi R^2 \Rightarrow$  A

Label D
Print "AREA"
Print A
Wait
Goto M

Label Q
ClrT
Print " "
Print "    PRESS HOME KEY"
```

**Exercise 3****Program QUAD**

```
DrawOFF
PlotOFF
ClrG
"AX2 + BX + C" ⇒ Y1
Print "    QUADRATIC EQUATION"
Print "    AX2 + BX + C = 0"
Input A
Input B
Input C
B2 - 4AC ⇒ D
If D < 0
Then
Print "COMPLEX ROOTS"
End
EndIf
(-B+√D)÷2A ⇒ R
(-B-√D)÷2A ⇒ S
Print "ROOTS"
Print R
Print S
Wait
If R < S
Then
R - 2 ⇒ Xmin
S + 2 ⇒ Xmax
Else
S - 2 ⇒ Xmin
R + 2 ⇒ Xmax
EndIf
0 ⇒ Xscl
Zm_Auto
Wait
```

Press **ENTER** to exit from the graph. Once the program has finished, you can press **TRACE** or change the WINDOW and re-graph.

## 11 Keystrokes used in the programs

[	$\boxed{2\text{ndF}} \boxed{\text{STAT}} \text{ (MATRIX)} \boxed{\text{E}} \boxed{1}$
]	$\boxed{2\text{ndF}} \boxed{\text{STAT}} \text{ (MATRIX)} \boxed{\text{E}} \boxed{2}$
:	$\boxed{\text{ALPHA}} \boxed{\text{Exp}}$
"	$\boxed{\text{PRGM}} \boxed{\text{A}} \boxed{2}$
=	$\boxed{\text{ALPHA}} \boxed{(-)}$
<	$\boxed{\text{MATH}} \boxed{\text{F}} \boxed{5}$
$\pi$	$\boxed{2\text{ndF}} \boxed{(-)}$
$\square$ (space)	$\boxed{\text{ALPHA}} \boxed{\square}$
$\sqrt{\quad}$	$\boxed{2\text{ndF}} \boxed{\square}$
$\Rightarrow$	$\boxed{\text{STO}}$
AxesOff	$\boxed{2\text{ndF}} \boxed{\text{ZOOM}} \text{ (FORMAT)} \boxed{\text{D}} \boxed{2}$
Circle(	$\boxed{2\text{ndF}} \boxed{\text{WINDOW}} \text{ (DRAW)} \boxed{\text{A}} \boxed{9}$
ClrG	$\boxed{\text{PRGM}} \boxed{\text{C}} \boxed{2}$
ClrT	$\boxed{\text{PRGM}} \boxed{\text{C}} \boxed{1}$
<i>colon</i>	$\boxed{\text{ALPHA}} \boxed{\text{Exp}}$
dim(	$\boxed{2\text{ndF}} \boxed{\text{STATPLOT}} \text{ (LIST)} \boxed{\text{A}} \boxed{3}$
Draw	$\boxed{2\text{ndF}} \boxed{\text{WINDOW}} \text{ (DRAW)} \boxed{\text{A}} \boxed{6}$
DrawOFF	$\boxed{2\text{ndF}} \boxed{\text{WINDOW}} \text{ (DRAW)} \boxed{\text{C}} \boxed{2}$
Else	$\boxed{\text{PRGM}} \boxed{\text{B}} \boxed{0} \boxed{5}$
End	$\boxed{\text{PRGM}} \boxed{\text{A}} \boxed{6}$
EndIf	$\boxed{\text{PRGM}} \boxed{\text{B}} \boxed{0} \boxed{6}$
For	$\boxed{\text{PRGM}} \boxed{\text{B}} \boxed{0} \boxed{7}$
Goto	$\boxed{\text{PRGM}} \boxed{\text{B}} \boxed{0} \boxed{2}$
If	$\boxed{\text{PRGM}} \boxed{\text{B}} \boxed{0} \boxed{3}$
Input	$\boxed{\text{PRGM}} \boxed{\text{A}} \boxed{3}$
L1	$\boxed{2\text{ndF}} \boxed{1}$
L2	$\boxed{2\text{ndF}} \boxed{2}$

Label	<code>PRGM</code> <code>B</code> <code>0</code> <code>1</code>
Next	<code>PRGM</code> <code>B</code> <code>0</code> <code>8</code>
PlotOFF	<code>PRGM</code> <code>G</code> <code>5</code>
Plt1(	<code>PRGM</code> <code>G</code> <code>1</code>
Print	<code>PRGM</code> <code>A</code> <code>1</code>
RclPict	<code>2ndF</code> <code>WINDOW</code> (DRAW) <code>F</code> <code>2</code>
Rg_ $x^2$	<code>STAT</code> <code>D</code> <code>0</code> <code>4</code>
Scatter $\square$	<code>STATPLOT</code> <code>G</code> <code>3</code>
Shade(	<code>2ndF</code> <code>WINDOW</code> (DRAW) <code>A</code> <code>7</code>
<i>space</i>	<code>ALPHA</code> <code>.</code>
StoPict	<code>2ndF</code> <code>WINDOW</code> (DRAW) <code>F</code> <code>1</code>
Text(	<code>2ndF</code> <code>WINDOW</code> (DRAW) <code>A</code> <code>0</code>
Then	<code>PRGM</code> <code>B</code> <code>0</code> <code>4</code>
Wait	<code>PRGM</code> <code>A</code> <code>4</code>
While	<code>PRGM</code> <code>B</code> <code>0</code> <code>9</code>
Wend	<code>PRGM</code> <code>B</code> <code>1</code> <code>0</code>
Xscl	<code>2ndF</code> <code>X/<math>\theta</math>/T/<math>n</math></code> (VARS) <code>B</code> <code>ENTER</code> <code>3</code>
Y1	<code>2ndF</code> <code>X/<math>\theta</math>/T/<math>n</math></code> (VARS) <code>A</code> <code>ENTER</code> <code>1</code>
Yscl	<code>2ndF</code> <code>X/<math>\theta</math>/T/<math>n</math></code> (VARS) <code>B</code> <code>ENTER</code> <code>6</code>
Zm_Auto	<code>ZOOM</code> <code>A</code> <code>1</code>
Zm_Default	<code>ZOOM</code> <code>A</code> <code>5</code>
Zm_Square	<code>ZOOM</code> <code>A</code> <code>6</code>
Zm_Stat	<code>ZOOM</code> <code>A</code> <code>9</code>