How Do I CODE on a CASIO fx-CP400

Making it possible for beginners

Includes pre-written code that you modify in supportive exercises


```
ViewWindow -7.7,7.7,1,-4.4,4.4,1
SetAxes Off
SetGrid Off
SetLabel Off
SetCoord Off

Lbl A

rand(0,6)⇒x
rand(-7,7)⇒a
rand(-4,4)⇒b
rand(-7,7)⇒c
rand(-4,4)⇒d

If x=0
 Then
 Line a,b,c,d,ColorBlack
IfEnd

If x=1
 Then
 Line a,b,c,d,ColorBlue
IfEnd

If x=2
 Then
 Line a,b,c,d,ColorRed
IfEnd

If x=3
 Then
 Line a,b,c,d,ColorMagenta
IfEnd

If x=4
 Then
 Line a,b,c,d,ColorGreen
IfEnd

If x=5
 Then
 Line a,b,c,d,ColorCyan
IfEnd

If x=6
 Then
 Line a,b,c,d,ColorYellow
IfEnd

Goto A
```

Team Steps

Made in Australia

How Do I CODE
on a CASIO fx-CP400

1st Edition.

First published in 2019.

Questions about this publication should be directed to support@stepsinlogic.com

ISBN 978-0-9807619-9-3

Art by Taryn.

Contents

1. Before your start

In this book you will execute pre-written lines of code (programs) and modify that code to make the program do as you want it too.

Thus you should first load the pre-written programs that are referred to in this book into either your fx-CP400 hand-held or your ClassPad Manager (for ClassPad II Series) software.

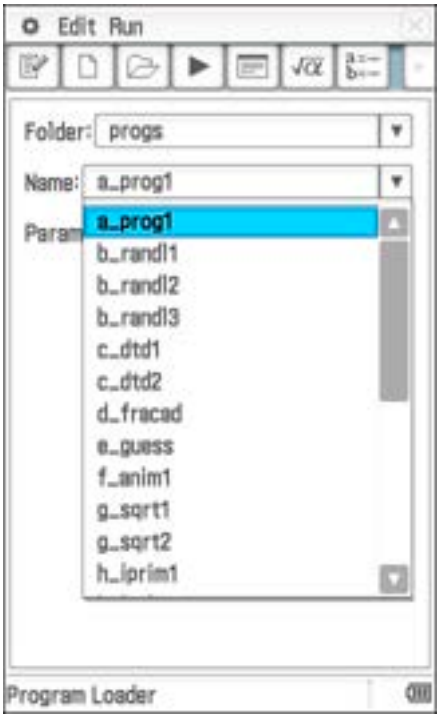
1.1 Loading the programs into the fx-CP400 (hand-held)

1. Visit https://casioeducation.com.au/software-emulators
2. Download the file **progs4_CP400_book.vcp** and save it to your chosen location.
3. Connect your fx-CP400 to your computer via USB and tap on USB Flash.
4. The flash memory drive of your fx-CP400 will mount on your computer.
   Copy the file progs4_CP400_book.vcp into the AutoImport folder on the flash memory drive of your fx-CP400.

5. Disconnect the flash memory drive of your fx-CP400 from your computer.

Upon disconnecting the flash memory drive of your fx-CP400, the programs will be automatically imported into a folder called progs. To see the programs:

6. Launch the Program application
7. Choose the folder progs
8. Tap the drop-down arrow of the Name list.



1.2 Loading the programs into the ClassPad Mangaer (software)

1. Visit https://casioeducation.com.au/software-emulators
2. Download the file **progs4_CP400_book.vcp** and save it to your chosen location.
3. Double-click on the file progs4_CP400_book.vcp and it will open in the ClassPad Manager.
4. To see the programs, complete steps 6 to 8 as seen above.

Note
Right-click (or Control-click) on the ClassPad Manager and look under Recent Documents to see that progs4_CP400_book.vcp is now number 1, or the open .vcp file. If you had content saved, prior to opening this file, it will not be accessible in this file. Choosing number 2, in the Recent Documents list, will re-open the .vcp file that was previously open.

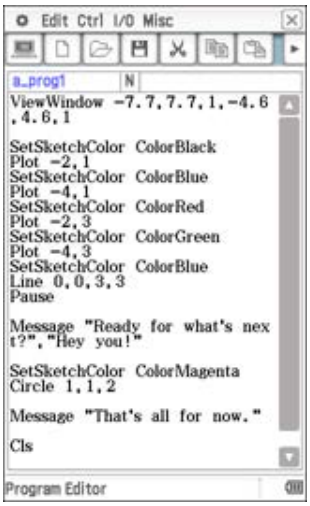
2. The basics of a program

Open the program application and the Program Loader screen is seen.

Choose the folder named progs.

The first program in the list is named a_prog1.

Tapping the script icon,  , shows the lines of code in the program that has been selected.



Tapping the computer icon,  , will return you to the Program Loader screen.

2.1 Understanding the code of a_prog1

The program is repeated below:

```
ViewWindow -7.7,7.7,1,-4.6,4.6,1

SetSketchColor ColorBlack
Plot -2,1
SetSketchColor ColorBlue
Plot -4,1
SetSketchColor ColorRed
Plot -2,3
SetSketchColor ColorGreen
Plot -4,3
SetSketchColor ColorBlue
Line 0,0,3,3
Pause

Message "Ready for what's
next?","Hey you!"

SetSketchColor ColorMagenta
Circle 1,1,2

Message "That's all for now."

Cls
Stop
```

Each line of code tells the fx-CP400 to do something when the program is executed.

In this program, the calculator draws various objects on the Cartesian plane.

`ViewWindow xmin,xmax,scale,ymin,ymax,scale` 'scales' the plane.

`Plot x,y` plots a *point*, at a specified location on the Cartesian plane, (-2,1) for example.

`Line a,b,c,d` plots a line between two specified locations on the Cartesian plane, (0,0) and (3,3) in this case.

Pause causes the program to pause, until the user commands it to continue. When paused, ▶ is seen at the bottom right of screen. Tapping ▶ causes the pause to cease.

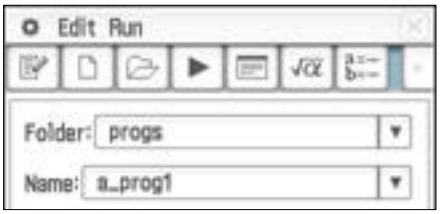
`Message "text","text"` displays a message to the user.

`Circle x,y,r` plots a circle with centre at a specified location on the Cartesian plane and with specified radius, centre (1,1) and radius 2 in this case.

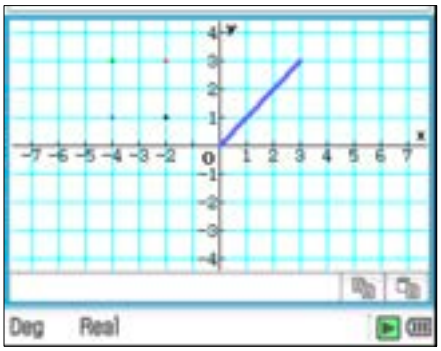
`Cls` clears all sketched objects from the screen.

`Stop` stops the program from executing. It is not necessary to include this command.

2.2 Running the code of a_prog1

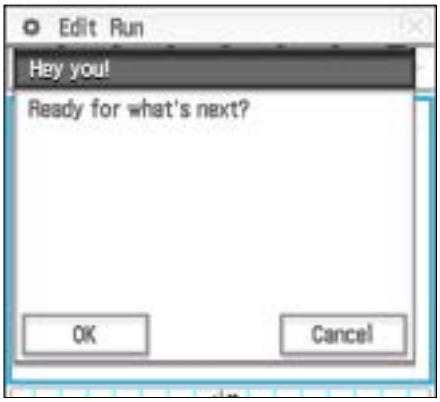


With a~prog1 selected in the Name field, tap ▶.



Four points are plotted and a line is drawn, and things seem to have ground to a halt!

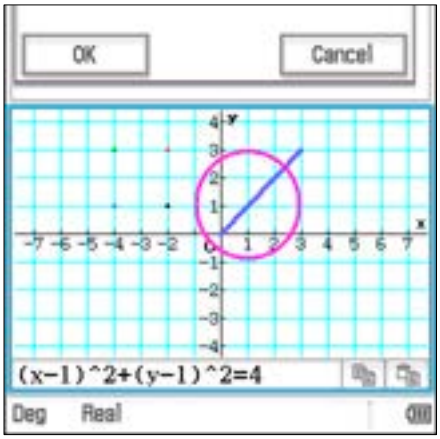
Notice ▶ bottom right of screen. This indicates the program has been paused by a command (Pause).



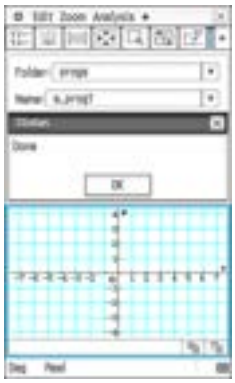
Tap ▶ and the program will continue.

A message is displayed. Which is a type of pause with extras.

Tap OK to continue.



A magenta circle is drawn and another message is displayed.
Tap OK.



The cartesian plane is cleared of all the objects that were drawn, and the program is Done.

Note that the line of code:

- `ViewWindow -7.7,7.7,1,-4.6,4.6,1`
- `SetSketchColor ColorBlack`

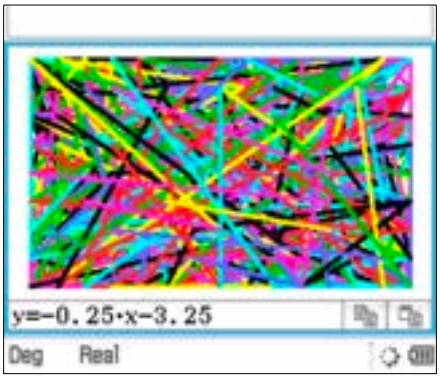
sets the view window settings and the sketching colour of the fx-CP400 *globally*. Thus, after running this program, if you draw a graph, for example, the ViewWindow will remain as set by the program and any sketches on that graph will be the last colour set by the program.

The ViewWindow can be changed by tapping , or selecting View Window from the menu. Sketch colour can be changed in the Graph Format setting also from the menu

If you did not see the axes/grid, as seen above, it will because of the settings on your machine in Graph Format. The program did not effect change on these settings, as it did the ViewWindow settings. You will see how to control these settings from within a program soon.

3. Breaking, If-Then-IfEnd, Lbl-Goto & variables

With b_randl2 selected in the Name field, tap ▶.



It seems to go on forever! To break (terminate) a program, at any time press Clear on the keyboard or tap Esc at the bottom of the screen. Pressing ← on the keyboard will pause the program.

Tap OK, as prompted, and the lines of code of the program are shown.

Tapping the computer icon, , will return you to the Program Loader screen.

How are all those colourful lines being produced, and how is their position determined?

```
Cls
ViewWindow -7.7,7.7,1,-4.6,4.6,1
SetAxes Off
SetGrid Off
SetLabel Off
SetCoord Off

Lbl A

rand(0,6)⇒x
rand(-7,7)⇒a
rand(-4,4)⇒b
rand(-7,7)⇒c
rand(-4,4)⇒d

If x=0
   Then
   Line a,b,c,d,ColorBlack
IfEnd

If x=1
   Then
   Line a,b,c,d,ColorBlue
IfEnd

If x=2
   Then
   Line a,b,c,d,ColorRed
IfEnd

If x=3
   Then
   Line a,b,c,d,ColorMagenta
IfEnd

If x=4
   Then
   Line a,b,c,d,ColorGreen
IfEnd

If x=5
   Then
   Line a,b,c,d,ColorCyan
IfEnd

If x=6
   Then
   Line a,b,c,d,ColorYellow
IfEnd

Goto A
```

The basic structure of this program is:

1. Setup various things
2. Place a Lbl (label) to 'go (back) to' at some point.
3. Define 5 variables, x, a, b, c and d.
4. Make seven If–Then-IfEnd statements based on the value of x.
5. Draw a line, the colour of which is determined by the value of x.
6. Go back (Goto) to Lbl 1

The process continues ad infinitum or until you break it or until the batteries give up. ☺

Note that Goto statements are used in this book to reduce the cognitive load for beginning programmers.

Exercise 1

In the program list you will be able to find programs named b_randl*, and randc*.

Execute each of them and see how they behave.
Predict how the code will be different to that seen opposite.

Examine the code of each and see if you were correct.

In this program we have chosen to define the colour of the lines using a different method than that used in a_prog1.

- `SetSketchColor ColorBlack` , as seen in a_prog1, is a 'global' function.
- `Line a,b,c,d,ColorGreen` , as seen in b_randl2, is a 'local' colouring.

'Local' colouring results in faster running programs, but takes a more effort to enter, often many entries versus a few. All Sketch functions (e.g. `Line a,b,c,d`, `Circle x,y,r`, `Plot x,y`) can have a colour added to the end of the function, e.g. `Circle 1,1,2,ColorCyan.`

Note
Since this program **sets** Axes Off, etc., and these settings are global settings, when you use another application on the fx-CP400 immediately after running this program you may have to reset various settings in the application.

Use the ⚙ menu.

3.1 About entering the code for the program named b_randl2.

If you are wondering how all this code is entered, do not worry, you will not have to do that for a while! You will be modifying pre-entered code for many activities in this book.

So that you are aware, however, when starting from scratch, code is either entered by:

a) finding a function/command in a menu and entering it as a "block" and then typing in the numbers/letters of the argument(s) from the keyboard as required,

or

b) typing the whole lot, if you know where the spaces and capitals and …, need to be.

Functions and commands can also be found in the Catalog on the soft keyboard.

The location of some functions and commands within the menus at the top of the screen, when the Program Editor is open, is given below.

`Cls`
I/O - Clear

`ViewWindow xmin,xmax,scale,ymin,ymax,scale`
Misc – Graph&Table(1)

`Set commands`
Misc – Setup(1) to Setup(4)

`Lbl-Goto`
Ctrl - Jump

`rand(a,b)`
Catalog (on soft keyboard)

`If-Then-IfEnd`
Ctrl - If

`SetSketchColor colour`
Misc – Setup(4)

`ColorRed`
I/O - color

`Line a,b,c,d`
I/O - Sketch

`Plot x,y`
I/O - Sketch

`Circle x,y,r`
I/O - Sketch

`Pause`
Ctrl - Control

`Message`
I/O – Output

`Stop`
Ctrl - Control

If you are a seasoned programmer and are using the ClassPad Manager (software), using the menus/catalog to enter code is a slow process, typing is far quicker, but can be less accurate. Once you learn the syntax, however, it is the best way. If programming on a Classpad hand-held, a mixture of approaches seems to work well.

For now, it is time to borrow another person's code and modify it! ☺

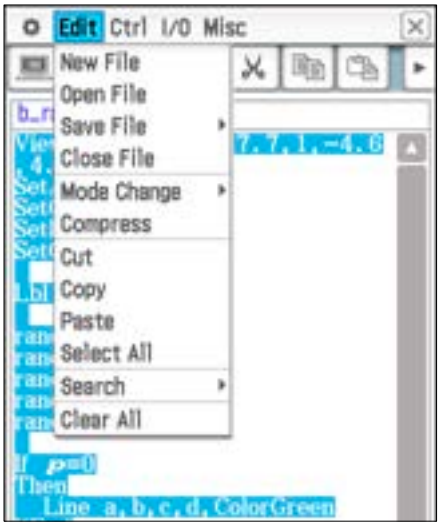
4. Borrowing and modifying code

Launch the Program application.

Tap  to create an empty (new) program file.

Type in the name Myfirst and press OK.

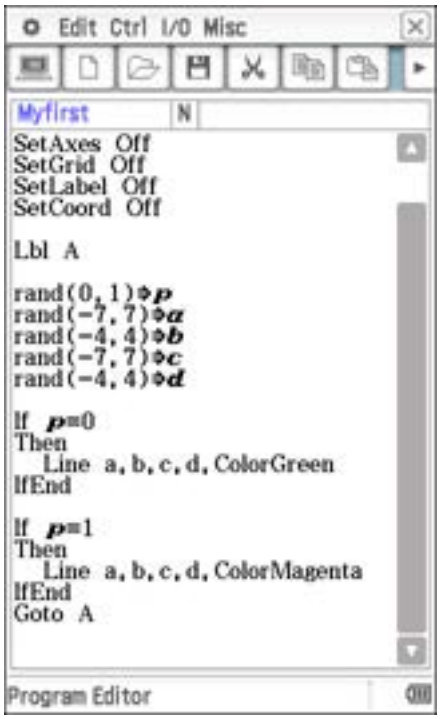
You are now ready to borrow some code.



Tap  and save the changes.
Open b_randl1

We want to copy all the lines of code in this program.
From the Edit menu choose Select All.

Then, from the Edit menu choose Copy.



Tap  and save the changes.

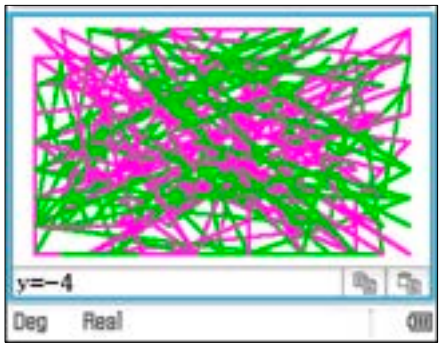
Open Myfirst.

Then, from the Edit menu choose Paste.

You are now ready to modify this code! ☺

Exercise 2

The code you have copied into Myfirst will execute the same way as b~randl1.



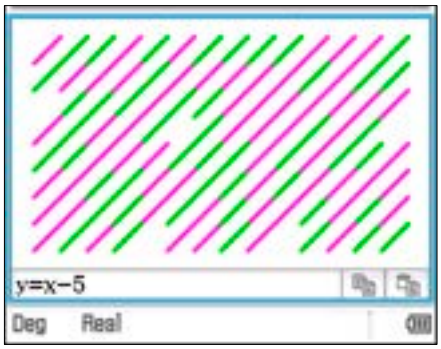
The start and end points of each line are not related (they are independent of each other). As such the lines are "all over the place".

Your task is to change the start and end points, so they are related in some way.

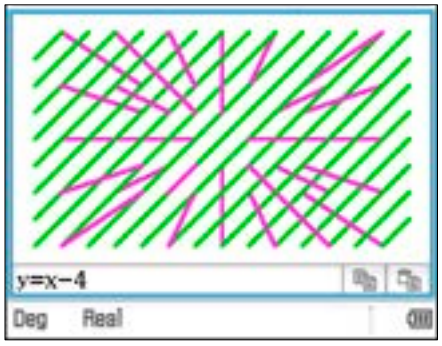
For example, not
`Line a,b,c,d ColorGreen`
but
`Line a,b,b,a ColorGreen`
or some other approach. You could try many different things, maybe:
`Line a,b,2a,b ColorGreen`

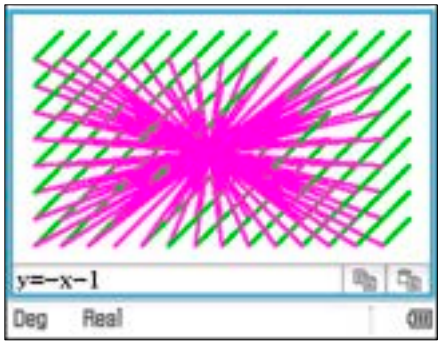
I tried various approaches and came up with the lovely art that follows.

First this:

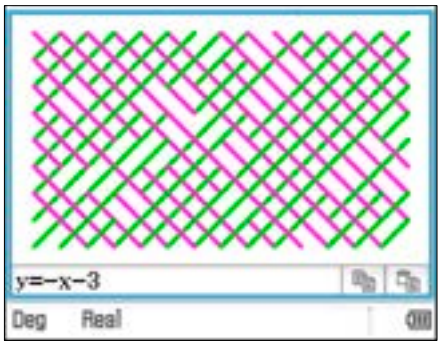


and then these:



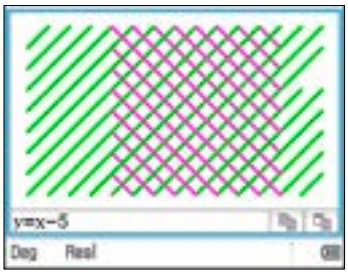


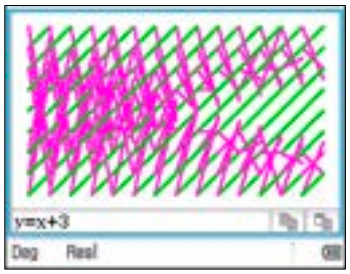
and then this:

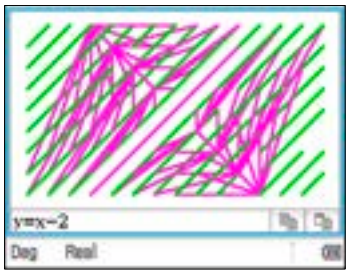


which is what I was aiming to do first but got those other nice patterns while making errors.

Here are some more I made.







What can you create?

Exercise 3

Make a new program called SCWORD.

SCWORD is short for screen-word (or showing a word on screen).

Use
`Line a,b,c,d`
Catalog (on soft keyboard)
and other commands to create a program that displays EAT! on screen, made from coloured lines.

You may like to start by using paper and pencil to draw a cartesian plane so you know where to start and end the lines.

Exercise 4

Repeat Exercise 3, but for a word of your choosing, complete with and exclamation mark.

Exercise 5

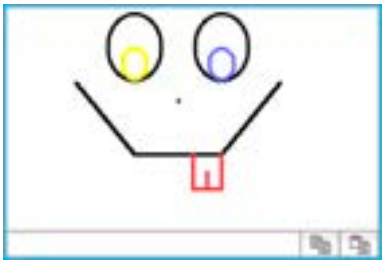
Repeat Exercise 4, but this time make the program display a flashing exclamation mark.

To do this draw the exclamation mark in a given colour and then re-draw it in a different colour and repeat this sequence over and over using:
`Lbl-Goto`
Ctrl - Jump
as seen in previous programs.

The "flashing" will be slower on a fx-CP400 hand-held compared to the ClassPad Manager software.

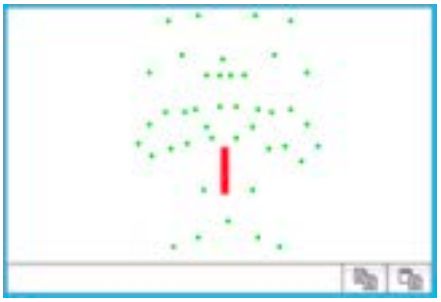
Once you have succeeded, add some other flashing elements to your creation.

You may want to graduate from words to a smiling face with a tongue hanging out – like some students have done! Below you can see different coloured eye-inners, with is part of the animation.



5. Dot-to-dot and Pause

Have you ever done a dot-to-dot drawing?
Can you imagine making an e-version of dot-to-dot but with a difference, given the machine will be doing most of the work. How do we ensure the element of surprise for the doer?



Run the program c_dtd1.

Note that the program *pauses* after drawing a red rectangle and some dots.

What could the drawing be, if the dots were joined with straight lines in some predetermined order?

To make the program continue, tap  .
Ooooo – scary!

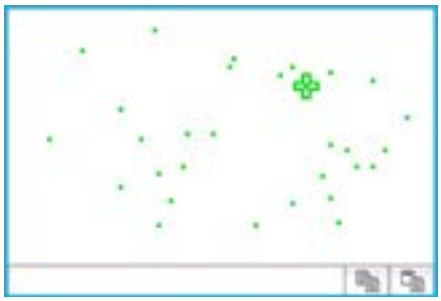
The complete program can be seen in Appendix 1.

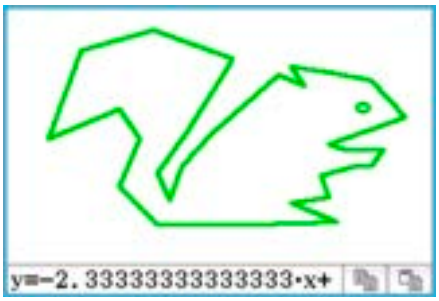
The *pause* is actioned the command:
`Pause`
Ctrl - Control

Below is an excerpt from the program.

```
SetSketchColor ColorGreen
.
.
.
Plot 0,1
Plot 0,2.1
Plot 0.4,2.1
PlotOff 10,10

Pause

SetSketchColor ColorBlack
Line 0,1,-0.5,2.1
Line -0.5,2.1,0.1,2.7
.
.
.
```

Having seen the above example, students were keen to create their own dot-to-dot.
Below you can see the dot part of one student's e-dot-to-dot, can you guess what it is?





Oh bless, a cute wee squirrel!

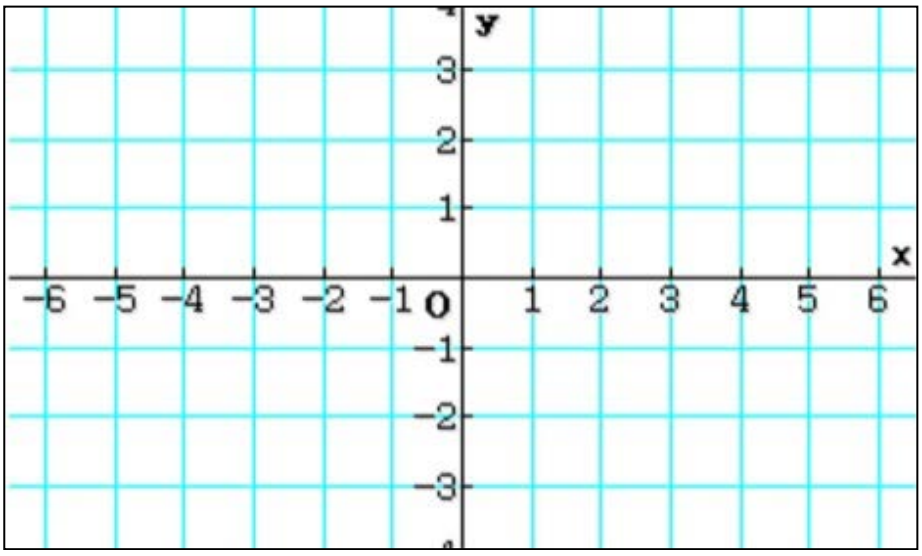
I wonder why his eye is like that. It is hardly round.

The student used the command `Circle 3.4,1.5,0.2` to draw the eye.
Can you imagine what the student has not done, assuming they wanted a circle?
Maybe it has something to do with this line from their program:
`ViewWindow -5,5,1,-4,5,1`

Exercise 6

Create your own e-dot-to-dot.

Start with a paper grid with axes drawn on it, something like that seen below, and then draw your 'critter' and carefully record the coordinates that form it.



Then start by writing at least the basic structure of the program on paper, before you start to enter it into the machine.

Once it is working, share your e-dot-to-dot with your friends.

Note
The colour in the program c_dtd1, seen in Appendix 1, is handled with the global command, e.g.
`SetSketchColor ColorBlack`
Since fast execution of code is not critical in this case, it is better (from a code entry point of view) to use a single line of code for 'colour definition'.

6. The While loop – adding on and on and on …

Consider following:

- $\frac{1}{4}$
- $\frac{1}{4}+\frac{1}{4\times4}$
- $\frac{1}{4}+\frac{1}{4\times4}+\frac{1}{4\times4\times4}$

Each of these are sequential steps in a growing string of numbers that are being added together. In each step we add on one number and we keep doing this forever. If the denominator of the next fraction always has one more $\times 4$ and the numerator is always 1, then the "forever" version can be written as follows:

$$\frac{1}{4}+\frac{1}{4\times4}+\frac{1}{4\times4\times4}+\frac{1}{4\times4\times4\times4}+\ldots\ldots+\frac{1}{4^n}$$

where n is a positive whole number.

Complete the following table for $1 \leq n \leq 8$, giving both fractional and decimal values.

n	1	2	3	4	5	6	7	8	. . .
sum (S)	$\frac{1}{4}$ 0.25	$\frac{5}{16}$ 0.3125							. . .

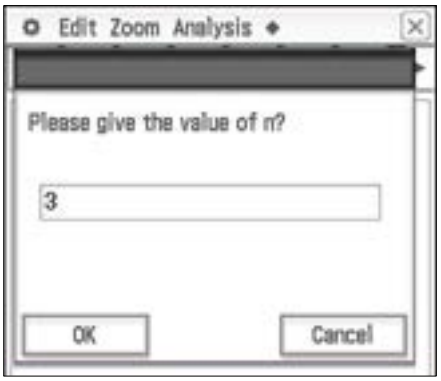
What do you notice?

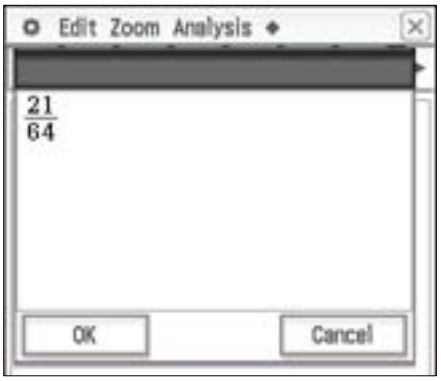
Can you suggest what will happen to the value of the sum, S, as the value of n becomes larger and larger (as $n \to \infty$)?

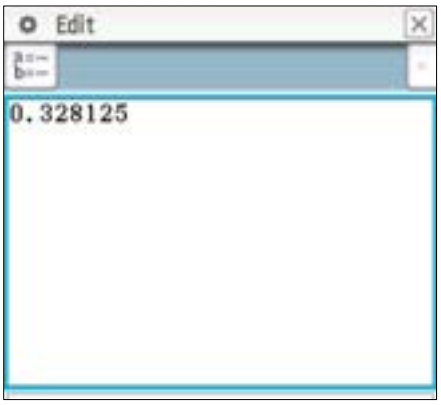
The program called d_fracad will add up a given number of steps.
You need to supply the number of steps, i.e. the value of n.

The program uses a While loop to add-on the next step. The program is shown below.

```
Local A,B,C,N
SetStandard
Cls

Lbl R
Cls

Input N,"Please give the value of n?"

0⇒A
4⇒B
1⇒C

While C≤N
  A+(1/B)⇒A
  B×4⇒B
  C+1⇒C
WhileEnd

SetStandard
PrintNatural A

SetDecimal
ClrText
Print A

Pause

Goto R
```







The calculator repeats the calculations in the While loop until C>N (while C≤N).

C is the variable that counts the number of times the calculations are done.

`While-WhileEnd` is located in:
Ctrl - While

`≤` is located in:
Ctrl – Logic (or on the Maths3 keyboard)

`SetStandard`
Misc – Setup(1)
commands the fx-CP400 to give an exact value output (fraction in this case).

`Input X,"Text"`
I/O – Input
defines variable and asks the user for its value.

`PrintNatural A`
I/O – Output

`SetStandard`
Misc – Setup(1)

`ClrText`
I/O – Clear

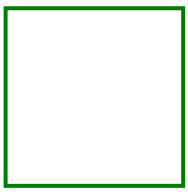
`Print X`
I/O – Output

`Local A,B,C,N`
Misc – Variable

The command Local defines the variables A,B,C and N, in this case, as local variables that are created when the program runs and deleted once the program ends. Using this approach means you do not end up with a lot of variables in the memory of the fx-CP 400 that serve no purpose.

If you look in the Variable Manager, found in the menu you will see variables that were defined from running the earlier programs, where Local function was not used.

Use d_fracad to compute the sum for larger values of n.
What appears to happen to the value of the sum as the value of n becomes larger and larger (as $n \to \infty$)?
Can you "fraction-up" the square, shown below, in a way that would convince another person that as $n \to \infty, S \to \frac{1}{3}$?



Exercise 7

Consider the sum:

$$S_n = \frac{1}{5} + \frac{1}{5\times5} + \frac{1}{5\times5\times5} + \ldots\ldots + \frac{1}{5^n}$$

Make a new program to compute the sum for chosen values of n.
I suggest you make a new program, then copy and paste the code from d_fracad into it and then modify! ☺

What happens to the value of S_n as $n \to \infty$?

Exercise 8

Consider the sum:

$$S_n = \frac{1}{6} + \frac{1}{6\times6} + \frac{1}{6\times6\times6} + \ldots\ldots + \frac{1}{6^n}$$

What happens to the value of S_n as $n \to \infty$?

Exercise 9

Consider the sum:

$$S_n = \frac{1}{7} + \frac{1}{7\times7} + \frac{1}{7\times7\times7} + \ldots\ldots + \frac{1}{7^n}$$

What happens to the value of S_n as $n \to \infty$?

Exercise 10

Consider the sum:

$$S_{a,n} = \frac{1}{a} + \frac{1}{a\times a} + \frac{1}{a\times a\times a} + \ldots\ldots + \frac{1}{a^n}$$

Can you suggest what happens to the value of $S_{a,n}$ as $n \to \infty$?

Can you prove that your suggestion is correct?

Exercise 11

Consider the sum:

$$S_n = 1 + \frac{2}{3} + \frac{4}{9} + \frac{8}{27} + \ldots\ldots + \frac{2^{n-1}}{3^{n-1}}$$

What happens to the value of S_n as $n \to \infty$?

7. If-Then-Else-IfEnd within a While loop. Can you guess my number?

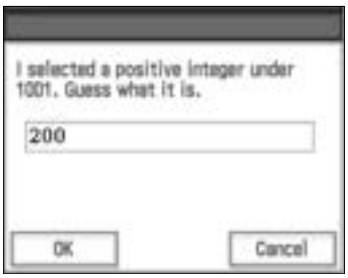
The program named e_guess, randomly selects a positive integer from the set {1,2,3, …, 1000} and prompts you to guess the integer.

When you enter your guess, the program determines if your guess was greater than or less than the positive integer it selected.

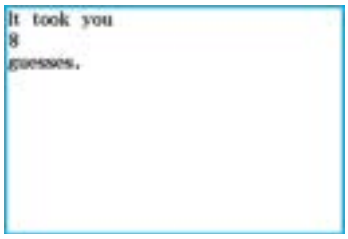
You are then prompted to guess again, and so on, until you enter a correct guess.

The program will then report how many guesses you made.

Play this game a few times. Play a few more times and see how good you can become.



I kept guessing …



Exercise 12

Before turning the page to look at the code in e_guess, sketch out what you think the program is doing and what commands might be required.

The code below is the main part of e_guess.

```
Local s,g,n
ClrText
0⇒n
rand(1,1000)⇒s

Input g,"I selected a positive integer under 1001. Guess what it is."


While g≠s

  1+n⇒n

  If g<s
    Then
      Message "Your guess is LESS than my selected positive integer."
    Else
      Message "Your guess is MORE than my selected positive integer."
  IfEnd

  Input g,"What is your guess now?"

WhileEnd


Message "You got it!"
Print "It took you"
Print n
Print "guesses."
```

`rand(a,b)` can be located in:
catalog (on soft keyboard)

While G≠S, the `If-Then-Else-IfEnd` loop repeatedly reports if the current guess is too small or too large.

The program pauses and waits for you to guess, thanks to the two *input* command lines:
`Input g,"I selected a positive integer under 1001. Guess what it is."`
`Input g,"What is your guess now?"`

`Input X,"Text"` can be located in:
I/O - Input

The line
`Print n`
prints the current value of n.

`Print` can be located in:
I/O - Output

Exercise 13

A bug exists in this code. If the user guesses correctly on their first guess, the program does not quite work correctly. Study the code and determine what happens. Squish the bug!

The complete form of e_guess is given below.
The additional lines, to those shown on the previous page, are shown in bold.

```
Local s,g,n
ClrText
0⇒n
rand(1,1000)⇒s

Lbl again

Input g,"I selected a positive integer under 1001. Guess what it is."

If int(g)≠g
   Then
   Goto again
IfEnd

While g≠s

  1+n⇒n

  If g<s
    Then
    Message "Your guess is LESS than my selected positive integer."
    Else
    Message "Your guess is MORE than my selected positive integer."
  IfEnd

  Lbl again2

  Input g,"What is your guess now?"

  If int(g)≠g
    Then
    Goto again2
 IfEnd

WhileEnd

Message "You got it!"
Print "It took you"
Print n
Print "guesses."
```

The additional lines act as a catch, stopping the user entering a guess that is not an integer. Note there are no catches for negative number or numbers greater than 1000.

The command `int(g)` calculates the integer part of the current value G.
For example `int(2.34) = 2`
`int(x)` can be located in:
catalog (on soft keyboard)

Exercise 14

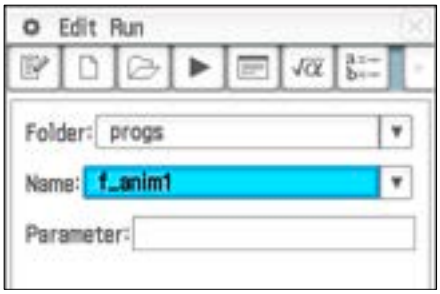
A simpler form of this guessing/checking process would be for the program to randomly choose the number 0 or 1 and then the player has to guess which number the program chose.

Make a program that performs this simpler guessing/checking process 5 times in a row and the program reports how many times the player guessed correctly.

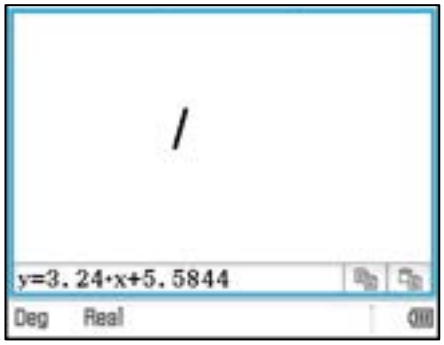
Once you succeed, modify your program so the player can set the number of times the guessing/checking process occurs.

8. Animation, using a For-Next loop

Run the program f_anim1.



While the image seen right does not show the action, it does display one 'frame' of the action.



The illusion of movement (animation) is created by the process of drawing an image, erasing that image and drawing another image nearby and then repeating this over and over.

Did you notice that the projectile changes length, as though it is elastic?

Here are the lines of code that produce the animation. Not too many lines at all.

```
Local x
ViewWindow -7.7,7.7,1,-4.4,4.4,1
SetAxes Off
SetGrid Off
SetLabel Off
SetCoord Off

Lbl A

Cls

For -3⇒x To 3 Step 0.2
  Line x,3x^2,x+0.4,3(x+0.4)^2,ColorBlack
  Cls
Next

Goto A
```

Reminder
Since this program **sets** Axes Off, etc., and these settings are global settings, when you use the fx-CP400 immediately after running this program you may have to reset the settings in the application you are using.

The heart of this program are the following lines of code.

```
For -3⇒x To 3 Step 0.2
  Line x,3x^2,x+0.4,3(x+0.4)^2,ColorBlack
  Cls
Next
```

The `Line` command is executed over and over again under the direction of the For-Next loop. The variable x controls how many times the For-Next loop is repeated and the position of the start and end of the line drawn.

`For-Next` can be located in:
Ctrl - For

Exercise 15

Change the value of 0.2 to 1, in the line:

```
For -3⇒x To 3 Step 0.2
```

Describe the effect of this change. Experiment with other values to create the opposite effect.

Exercise 16

Time to spice this program up a little.

a) Modify the program so the projectile is a colour other than black.
b) Modify the program so the projectile is a different colour on the way up to the way down.
c) Modify the program so the two projectiles are launched, one from left to right and one from right to left.

Exercise 17

Modify the program so the projectile's path is something other than a parabola.
For example, change 3-A^2 into 3-A.
Can you predict what change that will make to the projectile's path?
Experiment with other changes that give different paths.

Exercise 18

Run the program f_anim2.
Does it provide the illusion of 3-dimensional motion to you; a falling-spinning stick?

The code for f_anim2 can be found in Appendix 2.

Also included in the program set is a program called f_anim2, which results in the same animation as f_anim3, but it handles colouring the lines in the alternate manner.
Do you notice a difference in the speed of rotation?

9. Square roots, using a For-Next loop

Have you ever wondered how a calculator figures out a decimal approximation for the square root of a number – e.g. $\sqrt{24}$?

One way is for it to carry out a recursive process that can be described as follows:

- take a guess at the answer
- do a calculation using the guess
- the calculation gives a result that is more accurate
- use this result to do the calculation again and then
- keep repeating this over and over until the newest result is as accurate as required.

If we wish to calculate an approximation for $\sqrt{S}$, the guess is x_n and the first/next result is x_{n+1} then the calculation and process can be described as

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{S}{x_n}\right)$$

This process is the result of using Newton's Method to solve $x^2 - S = 0$.

You can read about it at https://en.wikipedia.org/wiki/Methods_of_computing_square_roots.

Before we use a program to do this, we will first do some calculations, so we gain an appreciation of the process.

Exercise 19

Let S =17 and let the guess, x_0, for the value of $\sqrt{17}$ be half of 17 (8.5).
One step in the recursive process is called an iteration.
Do 5 iterations.
Iteration one and two are given below.
Finish off the process.

Iteration 1:

$$x_1 = \frac{1}{2}\left(x_0 + \frac{S}{x_0}\right)$$

$$x_1 = \frac{1}{2}\left(8.5 + \frac{17}{8.5}\right) = 5.25$$

Iteration 2:

$$x_2 = \frac{1}{2}\left(x_1 + \frac{S}{x_1}\right)$$

$$x_2 = \frac{1}{2}\left(5.25 + \frac{17}{5.25}\right) = 4.244047619$$

Iteration 3:

$$x_3 = \frac{1}{2}\left(x_2 + \frac{S}{x_2}\right)$$

Compare your value for x_5 with that of other people.

4.123106 is the value of $\sqrt{17}$, correct to six decimal places. How close was x_5 to 4.123106?

The recursive (looping) nature of this process lends itself to computer automation.
A For-Next loop can be employed.

The program called g_sqrt1 automates this process.

In the previous exercise you were told to make the guess one-half of the value of S.
We use the same guess in this code.

The main part of the code is shown below.

```
ClrText
SetDecimal
Local x,i,I,S

Input S,"What square root do you
require?"
S/2⇒x

Input I,"How many iterations?"

For 1⇒i To I Step 1
   (1/2)*(x+(S/x))⇒x
   Print x
   Pause
Next

Stop
```

The heart of this program is the following lines of code.

```
For 1⇒i To I Step 1
   (1/2)*(x+(S/x))⇒x
   Print x
   Pause
Next
```

The calculation `(1/2)*(X+(S/X)) ⇒X`, is executed over and over again under the direction of the For-Next loop.
The variable I controls how many times the For-Next loop is repeated.

`For-Next` can be located in:
Ctrl - For

Run g_sqrt1 and calculate $\sqrt{17}$ with 5 iterations.

Check the output against the calculations you made in the previous exercise.

A partial output is shown opposite.

The complete g_sqrt1 program is shown below.

Note the bold lines, which were not shown on the previous page.
The extra code is a catch for an excessive number of iterations being entered by the user.

```
ClrText
SetDecimal
Local x,i,I,S

Input S,"What square root do you require?"
S/2⇒x

Lbl set_its
Input I,"How many iterations?"

If I>40
   Then
   Message "To conserve your taps, do less than 41. :)"
   Goto set_its
IfEnd

For 1⇒i To I Step 1
   (1/2)*(x+(S/x))⇒x
   Print x
   Pause
Next

Stop
```

Exercise 20

Suppose you want to find the $\sqrt{2\ 000\ 000\ 000}$.
Use g_sqrt1 to determine how many iterations are required to find $\sqrt{2\ 000\ 000\ 000}$ such that the 4th decimal place is correct.

Exercise 21

Experiment with numbers of the form 2×10^n to see whether or not a relationship exists between the number of iterations and n, for a given degree of accuracy.

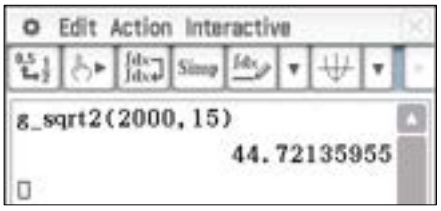
9.1 Programs as functions - g_sqrt2

g_sqrt2 does the same as g_sqrt1, but with less code, less output and it can run in both the Program application and the Main application.

S and I are defined to be the arguments/parameters of g_sqrt2, i.e. g_sqrt2(S,I).
This is done by entering them in the field top right.

If g_sqrt2 is run in Program, enter the values of S and I in the parameter field, e.g. 2000,15.

If run in Main, enter as shown below.



10. If-Then-IfEnd inside a For-Next; is that number prime?

Is 65 701 a prime number?

One way to figure out whether or not 65 701 is prime is to divide it by 2, 3, 4, 5, ….. , int($\sqrt{65\,701}$).
That is quite a bit of work and something a computer can do.

Advanced calculators and some mathematical software are equipped with a function called something like "isPrime", which simply states prime or not. The user, however, rarely gets to see how the function actually works, it is just black-box magic!

h_iprim1 is a program, shown below, that actions one algorithm that determines whether or not a positive integer, *n*, is prime.

The algorithm divides *n* by 2, then 3, then 4, then 5, then ….. and stops after reaching the integer value of $\sqrt{65\,701}$.

```
Local i,n

ClrText

Lbl integer

Input n,"Which integer do you want to test for primality ?"

If int(n)≠n
   Then
   Message "That is not an integer!"
   Goto integer
IfEnd

For 2⇒i  To int(√(n)) Step 1

   If int(n/i)=n/i
     Then
     Print n
     Print "is not prime."
     Print "It is composite."
    Goto end
   IfEnd

Next

Print n
Print "is prime."

Lbl end

Stop
```

Note the section in red print (above) is a catch for non-integer entries.

Also note that the If-Then-IfEnd block in green print lies inside the For-Next loop.
Thus the If-Then-IfEnd block is executed every time the For-Next loop loops.
In this program, the For-Next loop does not necessarily loop for the maximum possible number of times, as set by the value of n, because of the presence of the Goto jump.
The `Goto end` line, if actioned, will short cut the process, as we only need to find one number that divides n with zero remainder.

Exercise 22

Run h_iprim1 and try it out for numbers, **both prime and not prime** under 10,000.

Some primes under 10,000 are:
3659
6763
8429
9931

Be sure to try some composite numbers to check the program has no bugs.

Write down the time taken by the longest computation.

Exercise 23

Now try some larger primes but under 100,000. Here are three:

- 30011
- 68909
- 97579

Write down the time taken by the longest computation.

Exercise 24

What about 1246537?
Is it prime? How long did it take to determine whether or not it was?

Exercise 25

Test each of the following numbers for primness and record the time taken (t) by the program to determine whether or not the number was prime.

- 7
- 73
- 739
- 7393
- 73939
- 739391
- 7393913
- 73939133

Let l be the logarithm of the number being tested.
Draw a graph of l vs t.

Exercise 26

Run h_iprim1 again and this time test a negative number for primness, e.g. – 29 .
Describe what happened and why it happened.
Modify the program to include a catch for entering negative numbers.
You might find the absolute value function (`abs(x)` or | |) a useful tool.

`abs(x)` or | | can be located:
On the Maths1 soft keyboard
or in the Catalog (on the soft keyboard)

Exercise 27

h_iprim1, does not contain a fast algorithm.
If it determined that 2 did not divide N, then there is no point in dividing by all of the other even numbers, as it currently does.
Modify this program so that it avoids doing all those unnecessary divisions by even numbers.
Does this modification result in a speed improvement?

Exercise 28

Consider the program h_iprim2, shown below.
It employs a different algorithm than that used in h_iprim1.
Study the code and describe the different algorithm.
Determine whether or not the algorithm used is faster than the one used in h_iprim1 by using the numbers from Exercise 24.

```
Local i,n

ClrText

Lbl integer

Input n,"Which integer do you want to test for primality ?"

If int(n)≠n
  Then
  Message "That is not an integer!"
  Goto integer
IfEnd

For 2⇒i To 3 Step 1
   If int(n/i)=n/i
     Then
     Print n
     Print "is not prime."
     Goto end
   IfEnd
Next

For 1⇒i To int((√(n)+1)/6) Step 1
   If int(n/(6i-1))=n/(6i-1)
     Then
     Print n
     Print "is not prime."
     Goto end
   IfEnd
Next

For 1⇒i To int((√(n)+1)/6) Step 1
   If int(n/(6i+1))=n/(6i+1)
     Then
     Print n
     Print "is not prime."
     Goto end
   IfEnd
Next

Print n
Print "is prime."

Lbl end

Stop
```

Exercise 29

Study the program h_iprim3, shown below. It takes a different approach to h_iprim2.

```
Local n,i,k,d
ClrText
Lbl integer

Input n,"Which integer do you want
to test for primality ?"

If int(n)≠n
  Then
  Message "That is not an integer!"
  Goto integer
IfEnd

For 2⇒i To 3 Step 1
   If int(n/i)=n/i
     Then
     Print n
     Print "is not prime."
     Goto end
   IfEnd
Next

1⇒k
1⇒d

While d<int(√(n))

   6k-1⇒d

   If int(n/d)=n/d
     Then
     Print n
     Print "is not prime."
     Goto end
   IfEnd

   k+1⇒k

WhileEnd

1⇒k
1⇒d

While d<int(√(n))
   6k+1⇒d
   If int(n/d)=n/d
     Then
     Print n
     Print "is not prime."
     Goto end
   IfEnd

   k+1⇒k

WhileEnd

Print n
Print "is prime."

Lbl end
Stop
```

After studying the code, explain the different approach.

Try it out. Is it faster than h_iprim2? Document your findings.

Use h_iprim3 to test these two monsters:

- 591558727
- 5915587277.

Time accurately.

11. Whiles inside Ifs, (Whiles inside Ifs) inside Whiles – finding prime factors

The h_iprim* set of programs determines whether or not a positive integer is prime.
As such the algorithm needs to only check up to the square root of N.
But what if we wanted to determine the prime factors of N, how could this be done, particularly if N was a large number, either prime or composite ?

The program i_pfact determines the prime factors of positive integers.

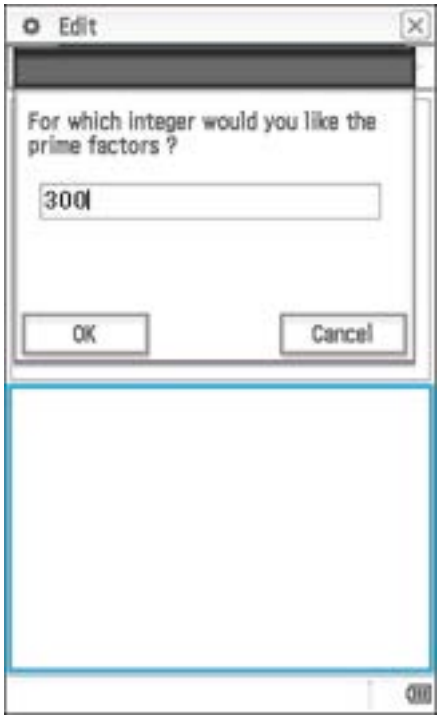
Exercise 30

Before looking at the code in i_pfact (next page), figure out how you would determine, without a program, all of the prime factors of 300.

Write out a clear set of steps that a young person, who only knows the basics of number, could understand.

Exercise 31

Run the program i_pfact.
Start with 'sanity' check with numbers – like 19, 25 and 50 – to be sure you are confident the program is working well enough. Then try 300.



Finally use i_pfact to find the prime factorisation of each of the following numbers.

- 8
- 74
- 740
- 2310
- 7394
- 73941
- 739413
- 7394133
- 73939133

Exercise 32

Study the i_pfact code below. Write down an explanation of how it works.
`frac(X)` is a command that determines the fraction part of X.
For example, `frac(12.8)` = 0.8

`frac(X)` can be located in:
Catalog (on the soft keyboard)

```
Local n,d

ClrText

Lbl integer

Input n,"For which integer would you like the prime factors ?"
2⇒d

If abs(int(n))≠n
  Then
  Message "That is not an positive integer!"
  Goto integer
IfEnd

Print "The prime factors are:"

If frac(n/d)=0
  Then

  While frac(n/d)=0
    Print d
    n/d⇒n
  WhileEnd

  3⇒d
  Else
  3⇒d
IfEnd

While d≤n

  If frac(n/d)=0
    Then

    While frac(n/d)=0
       Print d
       n/d⇒n
    WhileEnd

    d+2⇒d
    Else
    d+2⇒d

  IfEnd

WhileEnd

Print "That is all there is! :)"
Stop
```

Exercise 33

If you found i_pfact to be slow, especially as the input integers became larger, then you have a challenge. Develop a faster algorithm or google one and code it up.

12. Writing code in a text file and importing it to the fx-CP400

Entering code directly into a fx-CP400 handheld can be surprisingly easy if you are adept at finding all of the functions/commands and know the related syntax. See Appendix 4.

If you are a confident coder and have a good eye and high regard for correct syntax, it is possible to type the code on a computer, saving as a text file (.txt), and then transfer the text file to the fx-CP400 handheld.
The text file can be converted to the correct format, Normal, and then run on the fx-CP400.

Full details about this process can be found at:
https://support.casio.com/storage/en/manual/pdf/EN/004/ClassPadII_UG_EN.pdf
Section 12-1, page 201.

If using the ClassPad Manager software, choosing resizable mode and typing on the computer's keyboard makes for easy code entry.

13. Projects

Each of the following projects can be approached by creating a program that will do the repetitive parts of the work required.

Project 1 – Find the number

Find the smallest positive integer, n, for which $\sqrt{n} - \sqrt{n-1} < 0.01$.

Project 2 - DS

231 is a three-digit number.
The digit sum of 231 is 6.
How many three-digit numbers have a digit sum of 6?

How many three-digit numbers have a digit sum of 1?
How many three-digit numbers have a digit sum of 2?
How many three-digit numbers have a digit sum of 3?
…

You get the idea. Observe, wonder and make a conjecture. Answer the question, "Why?".

Project 3 - Sqtrinums

36 is both a square number and a triangle number. It is the smallest such number. Let such numbers be called sqtrinums.
1225 is the next smallest sqtrinum. Find the next four lowest sqtrinums.

Project 4 - Darts

Three darts are thrown at a rectangular board that is divided into three sections. If a dart lands in section 1 the player earns 8 points; 10 points are earned for section 2 and 12 points for section 3.

Assuming:

1) a shot is one throw of each dart
2) each dart lands in one of the three sections
3) the score for a shot is the sum of the points earned,

how many different scores are possible and what are they?

Project 5 - Riffle

One example of a card shuffling technique is called the riffle.
You can read about it at https://en.wikipedia.org/wiki/Shuffling#Riffle
How many riffles does it take to return each card in a pack to the position they had before the first riffle?
Does it depend?

Appendix 1 – c_dtd1

```
ViewWindow -7.7,7.7,1,-4.4,4.4,1
SetAxes Off
SetGrid Off
SetLabel Off
SetCoord Off

Lbl A

SetSketchColor ColorGreen
Cls

Line 0.1,-2,0.1,-0.5,ColorRed
Line 0.15,-2,0.15,-0.5,ColorRed
Line 0.2,-2,0.2,-0.5,ColorRed
Line 0.25,-2,0.25,-0.5,ColorRed

Plot 0,1
Plot -0.5,2.1
Plot 0.1,2.7
Plot 0.9,2.1
Plot 0.6,1
Plot 1.4,0.9
Plot 1.3,4.2
Plot 2,2.8
Plot 1.9,0.8
Plot 2.6,0.9
Plot 2.6,4
Plot 3.2,2.2
Plot 3.2,0.4
Plot 1.2,0.3
Plot 3.6,-0.4
Plot 2.2,-3.9
Plot 3,-0.9
Plot 2.4,-0.4
Plot 1.4,-3.6
Plot 1.8,-0.5
Plot 0.6,-0.1
Plot 1.2,-1.9
Plot 0.3,-3
Plot -0.6,-1.9
Plot -0.3,-0.1
Plot -1.2,-0.3
Plot -0.8,-3.6
Plot -1.8,-0.5
Plot -2.5,-0.7
Plot -1.7,-3.9
Plot -3,-0.3
Plot -0.5,0.3
Plot -2.6,0.4
Plot -2.6,2.2
Plot -1.9,4
Plot -2,0.8
Plot -1.3,0.8
Plot -1.4,2.8
Plot -0.8,4.2
Plot -0.9,0.9
Plot 0,1
Plot 0,2.1
Plot 0.4,2.1
PlotOff 10,10
Pause
```

```
SetSketchColor ColorBlack
Line 0,1,-0.5,2.1
Line -0.5,2.1,0.1,2.7
Line 0.1,2.7,0.9,2.1
Line 0.9,2.1,0.6,1
Line 0.6,1,1.4,0.9
Line 1.4,0.9,1.3,4.2
Line 1.3,4.2,2,2.8
Line 2,2.8,1.9,0.8
Line 1.9,0.8,2.6,0.9
Line 2.6,0.9,2.6,4
Line 2.6,4,3.2,2.2
Line 3.2,2.2,3.2,0.4
Line 3.2,0.4,1.2,0.3
Line 1.2,0.3,3.6,-0.4
Line 3.6,-0.4,2.2,-3.9
Line 2.2,-3.9,3,-0.9
Line 3,-0.9,2.4,-0.4
Line 2.4,-0.4,1.4,-3.6
Line 1.4,-3.6,1.8,-0.5
Line 1.8,-0.5,0.6,-0.1
Line 0.6,-0.1,1.2,-1.9
Line 1.2,-1.9,0.3,-3
Line 0.3,-3,-0.6,-1.9
Line -0.6,-1.9,-0.3,-0.1
Line -0.3,-0.1,-1.2,-0.3
Line -1.2,-0.3,-0.8,-3.6
Line -0.8,-3.6,-1.8,-0.5
Line -1.8,-0.5,-2.5,-0.7
Line -2.5,-0.7,-1.7,-3.9
Line -1.7,-3.9,-3,-0.3
Line -3,-0.3,-0.5,0.3
Line -0.5,0.3,-2.6,0.4
Line -2.6,0.4,-2.6,2.2
Line -2.6,2.2,-1.9,4
Line -1.9,4,-2,0.8
Line -2,0.8,-1.3,0.8
Line -1.3,0.8,-1.4,2.8
Line -1.4,2.8,-0.8,4.2
Line -0.8,4.2,-0.9,0.9
Line -0.9,0.9,0,1

Plot 0,2.1
Plot 0.4,2.1
PlotOff 10,10

Stop
```

Appendix 2 – f_anim2

```
Local x
ViewWindow -4,4,1,-12,1,1
SetAxes Off
SetGrid Off
SetLabel Off
SetCoord Off
SetRadian

Lbl A

Cls

For 0⇒x To 7 Step 0.2

  If x<1.7
    Then
    SetSketchColor ColorBlue
  IfEnd

  If (x≥1.7 and x<4.7)
    Then
    SetSketchColor ColorRed
  IfEnd

  If (x≥4.7 and x<6)
    Then
    SetSketchColor ColorBlue
  IfEnd

  Line cos(x),0.7sin(x)-2x,cos(x+3.14),0.7sin(x+3.14)-2x
  Cls

Next

Goto A
```

Appendix 3 – f_anim3

```
Local x
ViewWindow -4,4,1,-12,1,1
SetAxes Off
SetGrid Off
SetLabel Off
SetCoord Off
SetRadian

Lbl A

Cls

For 0⇒x To 7 Step 0.2

  If x<1.7
    Then
    Line cos(x),0.7sin(x)-2x,cos(x+3.14),0.7sin(x+3.14)-2x,ColorBlue
  IfEnd

  If (x≥1.7 and x<4.7)
    Then
    Line cos(x),0.7sin(x)-2x,cos(x+3.14),0.7sin(x+3.14)-2x,ColorRed
  IfEnd

  If (x≥4.7 and x<6)
    Then
    Line cos(x),0.7sin(x)-2x,cos(x+3.14),0.7sin(x+3.14)-2x,ColorBlue
  IfEnd

  Cls

Next

Goto A
```

Appendix 4 – Functions and commands, where to find them

The functions and commands used in the programs in this book can be entered by either:

- locating them in a menu when the Progam Editor is open, as outlined below, or
- locating them in the Catalog (on the soft keyboard) or
- typing them in, with correct syntax.

Some can also be located on the soft key boards tabs, like Math3.

Not all are located in the menus of the Program Editor.

`≤`
Ctrl - Logic or
Math3 on soft keyboard.

`abs(x)` or `| |` can be located:
Catalog (on the soft keyboard)
or on the Math1 soft keyboard

`Cls`
I/O - Clear

`ClrText`
I/O – Clear

`ColorRed`
I/O - color

`Circle x,y,r`
I/O - Sketch

`For-Next`
Ctrl - For

`frac(X)`
Catalog (on soft keyboard)

`Line a,b,c,d`
I/O - Sketch

`If-Then-IfEnd`
Ctrl - If

`int(X)`
Catalog (on soft keyboard)

`Input X,"Text"`
I/O – Input

`Lbl-Goto`
Ctrl – Jump

`Local A,B,C,N`
Misc – Variable

`≤`
Ctrl – Logic or Math3 on soft keyboard.

`Message`
I/O – Output

`Pause`
Ctrl - Control

`Plot x,y`
I/O - Sketch

`Print X`
I/O – Output

`PrintNatural A`
I/O – Output

`rand(a,b)`
Catalog (on soft keyboard)

`Set commands`
e.g. `SetStandard, SetDecimal, SetRadian, SetStandard`
Misc – Setup(1) to Setup(4)

`SetSketchColor colour`
Misc – Setup(4)

`Stop`
Ctrl - Control

`ViewWindow xmin,xmax,scale,ymin,ymax,scale`
Misc – Graph&Table(1)

`While-WhileEnd`
Ctrl - While

For further information see Chapter 8 of
http://support.casio.com/storage/en/manual/pdf/EN/004/fx-CG50_Soft_v320_EN.pdf

fx-CP400

For further support such as:

- “HOW TO” videos
- Free lesson ideas, with video support
- How to show the fx-CP400 on the big screen
- The Prime Schools PLUS program
- PC fonts for worksheet production

visit

www.casioeducation.com.au