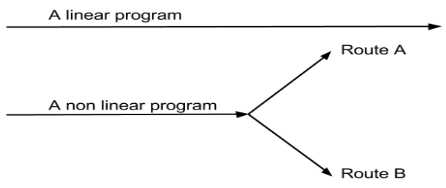
Making decisions

So far our programs are very linear, performing one operation after another; like a long straight road the destination is known. Our programs would be more interesting if we could introduce a few forks in the road, so the end point is not immediately obvious. We can achieve this by using a 'conditional statement'.



In the non-linear program the computer will take one of two paths depending on the condition at the fork. For example if this was a program for a central heating system the program may take route A if the room temperature was above 20 degrees centigrade or route B if the temperature was 20 degrees or below. Notice that we have naturally used the word ‘if’ to express the condition. In Small Basic and almost all other languages the IF/Then conditional statement can be used. Here is an example algorithm for a central heating system.

```
1. WHILE the central heating is ON
a. Measure the temperature
b. IF the temperature is equal to or
     more than 20° centigrade
     THEN switch off heating
c. IF the temperature is less than
     20° centigrade
     THEN switch on heating
2. ENDWHILE
```

This program will keep the temperature at 20° C.

In computer science the values True and False are known as Boolean data types after the mathematician George Boole.

When designing an algorithm that has a conditional statement it is important to understand that the condition must be able to evaluate to either 'True' or 'False'.

For example the question 'Is the temperature > 20 ?' clearly evaluates to either 'True' or 'False'. However, 'Is it a good film?', is too subjective and cannot always be reduced to a simple 'True'/'False' or 'Yes'/'No' answer.

Small Basic uses the comparison opertaors: greater than (>), greater than or equal to (>=), less than (<), less than or equal to (<=), same as (=) and, not equal to (<>).

In Small Basic these examples would evalute to 'True':

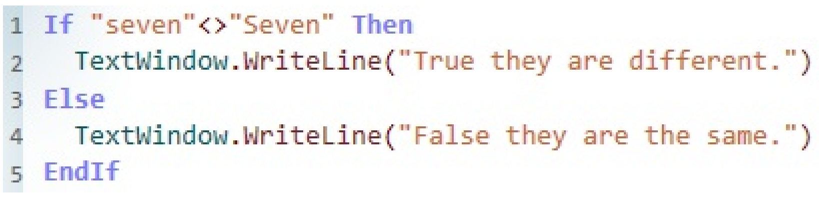
23 > 5
1.2 < 3.5
"seven" = "seven"
"April" <> "april"

and these examples would evaluate to 'False':

"Hal" = "hal"
129 > 400
23 >= 24

The (=) and (<>) comparison operators can be used with text.

Type the following code into the Small Basic editor. Before running the code, try to predict which statement will be printed to the 'TextWindow'.

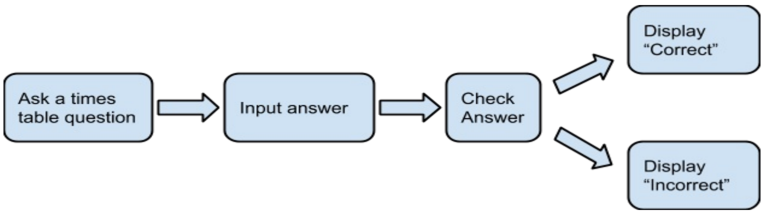


```
If "seven"<>"Seven" Then
  TextWindow.WriteLine("True they are different.")
Else
  TextWindow.WriteLine("False they are the same.")
EndIf
```

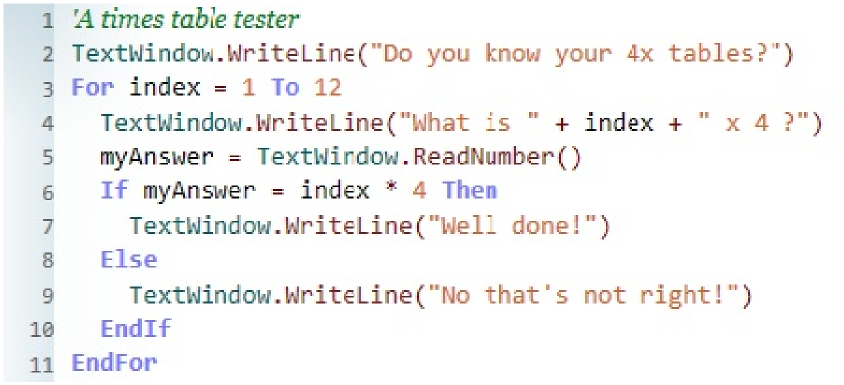
Using IF/THEN & ELSE statements

The WHILE loop that you used before utilised a conditional statement.

Small Basic uses simple IF/THEN and ELSE conditional statements to control decision processes in programs. The ELSE statment is used to define what the computer should do if the condition is not true. Let's write a simple times tables testing program to illustrate how these new keywords can be used. Below I have used a flowchart to plan our program, it is similar to our other algorithms but is pictorial. Drawing flowcharts can help you plan your program before writing it.



Here's the code written in small basic.



```
1  'A times table tester
2  TextWindow.WriteLine("Do you know your 4x tables?")
3  For index = 1 To 12
4    TextWindow.WriteLine("What is " + index + " x 4 ?")
5    myAnswer = TextWindow.ReadNumber()
6    If myAnswer = index * 4 Then
7      TextWindow.WriteLine("Well done!")
8    Else
9      TextWindow.WriteLine("No that's not right!")
10   EndIf
11 EndFor
```

This code uses the IF/THEN/ELSE keywords. The condition on line 6 is evaluated. If the result of the condition is TRUE (myAnswer is the same as index times 4) then "Well done!" is displayed. The ELSE statement is ignored and the loop goes around again for the next question.

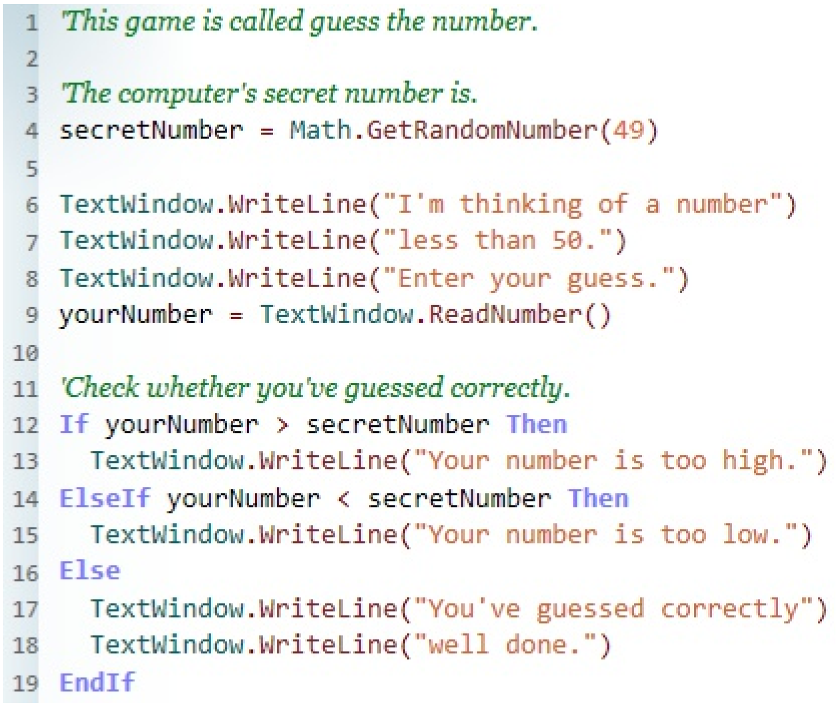
If the result of the condition is FALSE (myAnswer is not the same as index times 4) then the ELSE statement is executed and "No that's not right!" is displayed, then the next question is displayed.

A guess the number game

In this next example the program invites the user to guess a randomly generated number. The user is informed whether their guess was too high, too low or correct.

The code uses a new keyword, ELSEIF. This new keyword will only execute if the previous IF or ELSEIF statement evaluated to 'False'. It is possible to chain multiple ELSEIF statements together and as soon as one of them evaluates to 'True' then the others are skipped.

The ELSE statement defines a default behaviour; it only executes when all the others evaluate to 'False'.

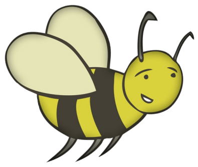


```
'This game is called guess the number.

'The computer's secret number is.
secretNumber = Math.GetRandomNumber(49)

TextWindow.WriteLine("I'm thinking of a number")
TextWindow.WriteLine("less than 50.")
TextWindow.WriteLine("Enter your guess.")
yourNumber = TextWindow.ReadNumber()

'Check whether you've guessed correctly.
If yourNumber > secretNumber Then
  TextWindow.WriteLine("Your number is too high.")
ElseIf yourNumber < secretNumber Then
  TextWindow.WriteLine("Your number is too low.")
Else
  TextWindow.WriteLine("You've guessed correctly")
  TextWindow.WriteLine("well done.")
EndIf
```



Activity 5: conditionals

1. Evaluate these following expressions to either 'True' or 'False'.

(a) 3.14 > 31.4

(b) 7 < 7

(c) 7 <= 7

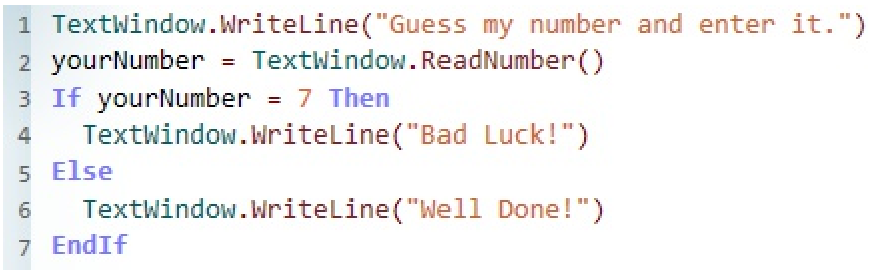
(d) "Tuesday" = "tuesday"

(e) 3 <> 3

(f) 798 <> 23

(g) 0.1 > 0.01

2. In the following code the computer is thinking of the number 7 and the user tries to guess which number it is. Can you identify the error in the program.



```
TextWindow.WriteLine("Guess my number and enter it.")
yourNumber = TextWindow.ReadNumber()
If yourNumber = 7 Then
  TextWindow.WriteLine("Bad Luck!")
Else
  TextWindow.WriteLine("Well Done!")
EndIf
```

3. The 'Guess the number' game only allows for one guess. Can you improve the program so that it allows you to make multiple guesses? You can adapt the program to give you either a fixed number of guesses or as many as you need. You'll need to use either a 'While' or 'For' loop.