

KS3 Programming Workbook

INTRODUCTION TO

BB4W

BBC BASIC **for** Windows

Name:

Class:

Resource created by Lin White

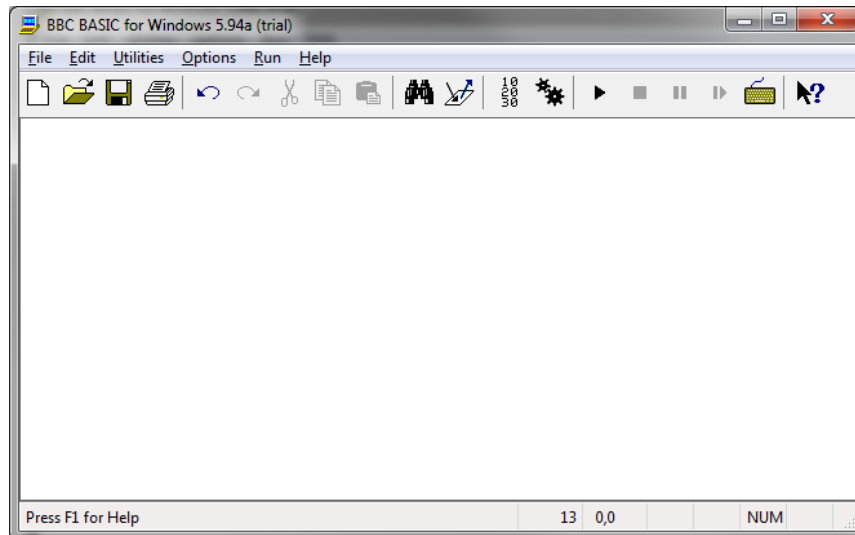
www.coinlea.co.uk

This resource may be photocopied for educational purposes

Introducing BBC BASIC for Windows

BASIC is a programming language that is easy to learn. It provides a way to write instructions that are simple for a human to understand and clear for the computer to follow. BBC BASIC was first provided on the BBC computers in the 1980s, and was the first programming language many children learned. This version, for Windows, is an up to date version that is still easy to learn, while allowing you to do many powerful things with your computer – and you're the one giving the orders!

Open up the BB4W program. You should see this:

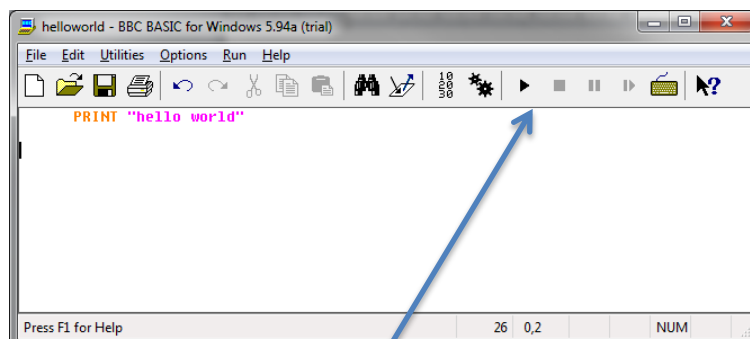


Your version might have (trial) written at the top or you might have the full version.

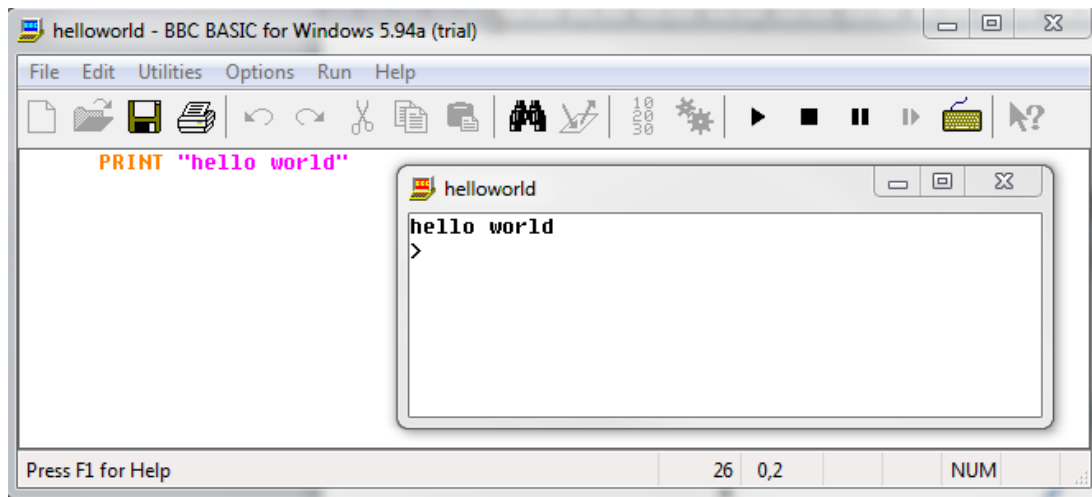
This looks very like a word processing program, except that what you type are instructions for the computer.

Type in `PRINT "hello world"`

You will need to type it in exactly as you see it here – type the word `PRINT` all in capitals. When you hit Enter to move to a new line, the program will colour code your writing.



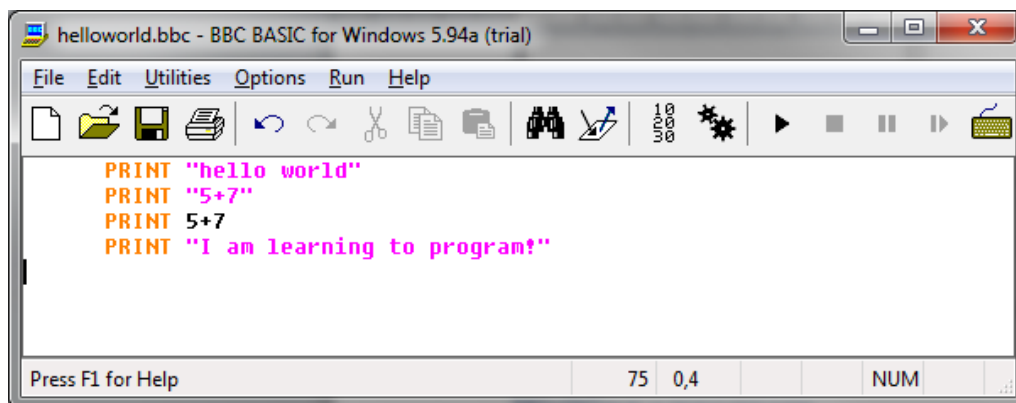
Now click the play button on the toolbar.



Note: I've resized all my program windows – yours will probably be much bigger.

Writing a program

So far you have only written one instruction. Close down the window that appeared, to get back to your BB4W window, and type in some more instructions.



The BB4W editor colours your code so that you can understand it more easily. Anything in pink is a **String**, which means BASIC treats it as text and will print it exactly as it appears. Anything that is in orange is a **Keyword**, which means it is a special word the computer understands as an instruction. Keywords are usually in capitals so they are easy to see.

Click on **File/Save as** and save your code with the filename **test**.

Click on the **Run** button again to see all your code running.

Did it work? Congratulations, you've written your first BASIC program!

Problems? If your program didn't run at all, check very carefully that you have typed it correctly and followed all instructions.

Programming time 1!

Now that you can write your very own program, write a program to solve these problems. Write down your answers. Can you explain what it does? Some are easy, some trickier.

Expression	Result	What does it do?
3+4		
57-4		
4*6		
7/2		
7MOD2		
7DIV2		
RND(10)		

Hint: Write a print statement, then a comma, then the expression you want to calculate.

For example:

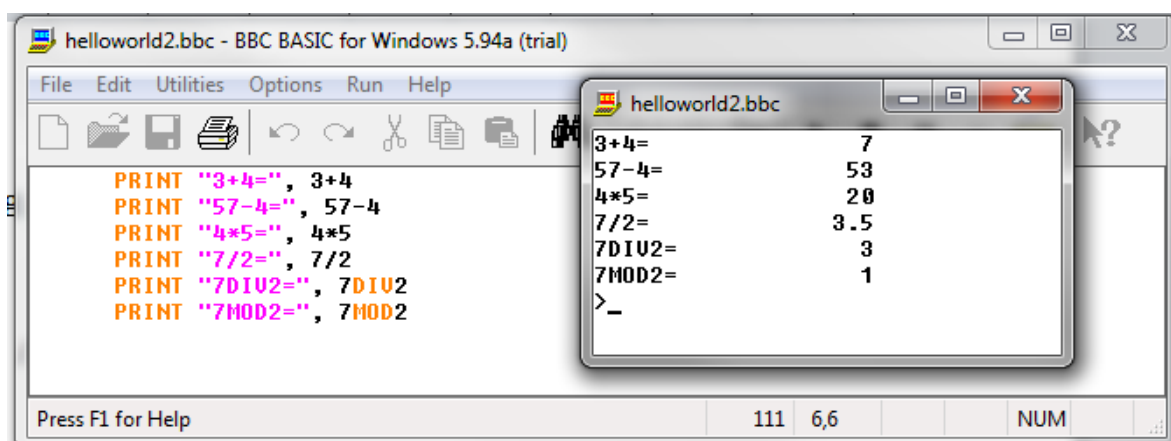
```
PRINT "3+4=", 3+4
```

What is the difference in using a ; (semi colon) instead of a , (comma)?

Challenge:

Write more statements using /, MOD and DIV. Can you work out what they do? They are all related.

Write more statements using RND (10) – can you work out what it does? Try it with different numbers instead of 10, to see if you are right.



The screenshot shows the BBC BASIC for Windows 5.94a (trial) interface. The main window displays a program with the following code:

```
PRINT "3+4=", 3+4
PRINT "57-4=", 57-4
PRINT "4*5=", 4*5
PRINT "7/2=", 7/2
PRINT "7DIV2=", 7DIV2
PRINT "7MOD2=", 7MOD2
```

A smaller window titled "helloworld2.bbc" shows the output of the program:

```
3+4=      7
57-4=     53
4*5=      20
7/2=      3.5
7DIV2=     3
7MOD2=     1
>_
```

The status bar at the bottom indicates "Press F1 for Help", "111", "6,6", and "NUM".

Problems?

It changes PRINT to print	Under Options, make sure lowercase keywords isn't ticked. Having the keywords in capitals makes them stand out more.
Can I make the print bigger?	Yes, go to Options, choose Set Font, take the tick out of Use System Fixed Font and then you can choose from a range of fonts and sizes. This will only affect your program listing, and not the output.
Can I change the colours?	Yes, but I recommend you stick with the standard colours until you're very confident, as they will help you spot errors in your code.
My 0 looks funny!	Some programming languages, and some programmers, display their 0s with a line through, to show it's a number and not the letter O.
What's the * for?	Computers can't tell the difference between x meaning times (multiply) and x that's a letter, so they need to use a different symbol for multiply. They use the * instead.

1234567890
1234567890
1234567890

Answers to Programming time 1

+	Adds the numbers
-	Subtracts (takes away)
*	Multiplies (times)
/	Divides – gives decimal part of answer as well
DIV	Gives just the whole part of the division answer
MOD	Gives just the remainder of the division answer
RND(x)	This will provide a random number between 1 and x (like rolling an x-sided dice). Very useful when writing games!

7/2=3.5 but you could also say 7/2 is **3 remainder 1**.

7DIV2=3

7MOD2=1 (the remainder is 1)

Using variables

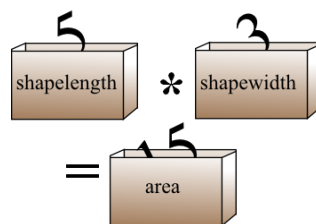
We can give a name to a value, so we can talk about it later. This is called a **variable**, because the value stored can vary.

PRINT $3*5$ doesn't tell you what the numbers are for.

```
shapelength=5
shapewidth=3
area=shapelength*shapewidth
PRINT area
```

Now you can see that the numbers are the length and width of a shape and the answer is the area. There's more to type this way but if we want to work out the area of a different size shape we can just change the numbers for shapelength and shapewidth and rerun the program. You'll see more reasons why a **variable** is useful later, but they are very important!

Using a **variable** with a name is like getting a box, writing the name on the side and then putting that number in the box. Next time the computer comes across the same name it will find the box and look up the number in it to use.



Programming time 2!

Write the program above and alter it and run it each time to calculate the area of a shape that is

1. length 3 and width 2
2. length 8 and width 5
3. length 445 and width 237

Write another program – it needs to calculate the total cost of items. You will need a variable for the item cost and a variable for the quantity. The totalcost needs to be $\text{itemcost} * \text{quantity}$.

Note: variable names can't have spaces in them. You can either run words together – `totalcost` – or use an underscore – `total_cost` – or make the first letter of the second word a capital – `totalCost`. You also can't have keywords at the start of your variable, which is why I called mine things like `shapelength` rather than `length`. You could just call them `L` and `W`, but in a long program you might forget what they mean.

Input in BASIC (numbers)

In order to write a useful program, we need to be able to ask the user questions. This means we need to store their answers as well.

We can ask someone their age and store it as a **variable** called age like this:

```
INPUT "What is your age" ; age
```

This stores the number entered in a box labelled age.

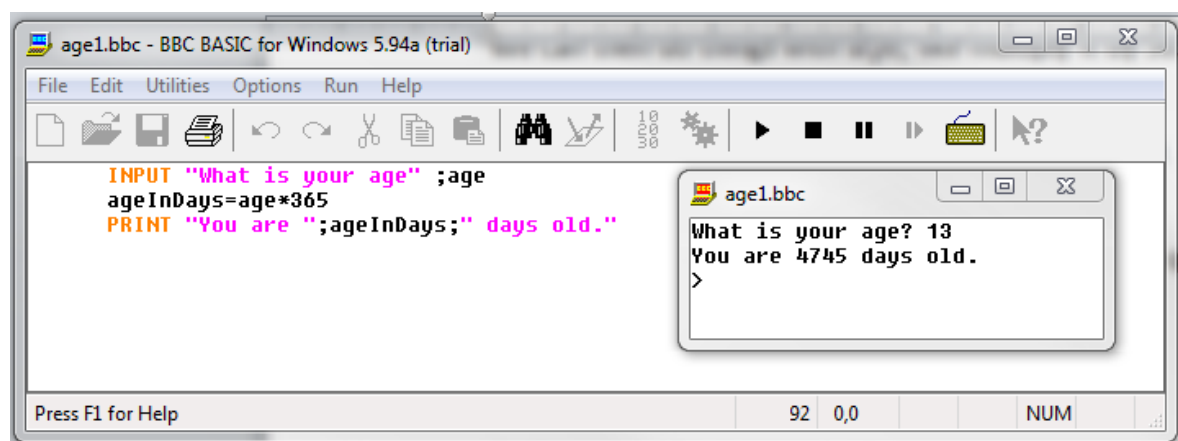
We can then do things with age, like multiply it by 365 to find out your age in days.

```
ageInDays=age*365  
PRINT "You are "; ageInDays ; " days old."
```

Note: you do not have to add a question mark to the question.

You do need to add spaces in where needed.

Be very careful with your capitals and spelling with variable names – how you do it isn't as important as doing it the same way each time you want to use it.



Programming time 3!

1. Write a program to work out your age in hours ($\text{age} * 365 * 24$).
The age 13 should produce 113880.
2. Write a program to ask for width and length and print out the area of a rectangle ($\text{shapewidth} * \text{shapelength}$).
Width 5 and height 2 should produce area 10.
3. Write a program to ask for time and speed and work out the distance travelled ($\text{time} * \text{speed}$).
Time 2 and speed 60 should produce distance 120.

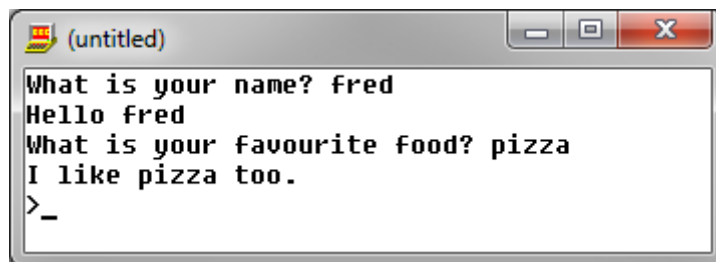
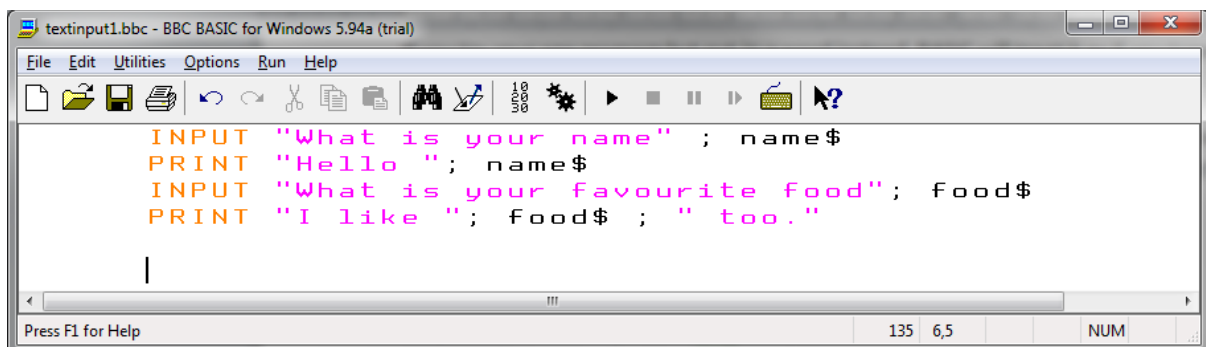
Make sure that each time you choose a sensible name for your variables.

Input in BASIC (Text)

If you try your age program but put in a word instead of a number, BASIC will treat it as if you typed in 0. This isn't much good if we want to type in text!

So far our variables have all been numbers, but if we want to store text instead, we need to put \$ on the end of the variable name. This is to remind us and the computer that it's storing a **String** (text) instead of a number.

```
INPUT "What is your name" ; name$
PRINT "Hello "; name$
INPUT "What is your favourite food"; food$
PRINT "I like "; food$ ; " too."
```



Programming time 4!

Write your own program to respond to the user. Test your program on a partner, and then help them test theirs.

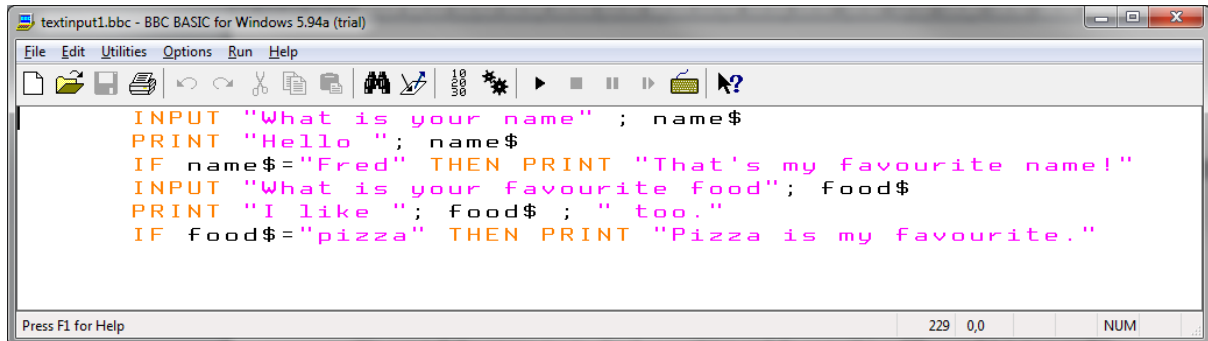
Remember to put \$ on the end of a variable if it is storing text. You can put % on the end if it is storing an integer (a whole number) or not put anything special on the end.

Variable ends in \$	Variable must contain text (can be numbers as well)
Variable ends in %	Variable must contain a whole number, no text
Variable has no special character at the end	Value must be a number (no text) but can have decimal places

Making choices in BASIC

We can tell the computer to check something and do something different if it is true or if it is false, by using an IF statement.

IF name\$="Fred" THEN PRINT "I like that name."



```
textinput1.bbc - BBC BASIC for Windows 5.94a (trial)
File Edit Utilities Options Run Help
INPUT "What is your name" ; name$
PRINT "Hello "; name$
IF name$="Fred" THEN PRINT "That's my favourite name!"
INPUT "What is your favourite food"; food$
PRINT "I like "; food$ ; " too."
IF food$="pizza" THEN PRINT "Pizza is my favourite."
Press F1 for Help 229 0,0 NUM
```

Note: You will need to use capitals and spelling exactly the same as you want them to type in. If you are checking for "Fred" and they type in fred the computer will not match the values!

We can use IF statements in the same way with numbers.

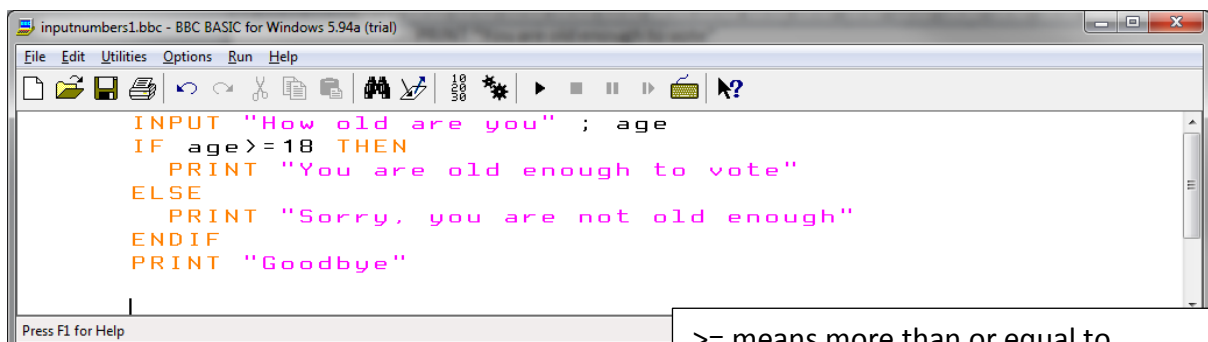
Note the way the code is laid out here, to make it easy to read.

```
IF age>=18 THEN
    PRINT "You are old enough to vote"
ELSE
    PRINT "Sorry, you are not old enough."
ENDIF
PRINT "Goodbye"
```

Start a new line after
THEN

Start a new line after
ELSE

put ENDIF at the end
of the actions affected
by the test



```
inputnumbers1.bbc - BBC BASIC for Windows 5.94a (trial)
File Edit Utilities Options Run Help
INPUT "How old are you" ; age
IF age>=18 THEN
    PRINT "You are old enough to vote"
ELSE
    PRINT "Sorry, you are not old enough"
ENDIF
PRINT "Goodbye"
Press F1 for Help
```

>= means more than or equal to.

In maths we would write \geq

Programming time 5!

1. Copy out the text input program and get it working. Copying out programs is a good way of learning how to sort out problems with code.
2. Develop the program to have a longer conversation with the computer. It should ask questions and then answer differently according to the answers given. Test your program out on a partner. Does it work?

Remember we use \$ at the end of a variable that stores text.

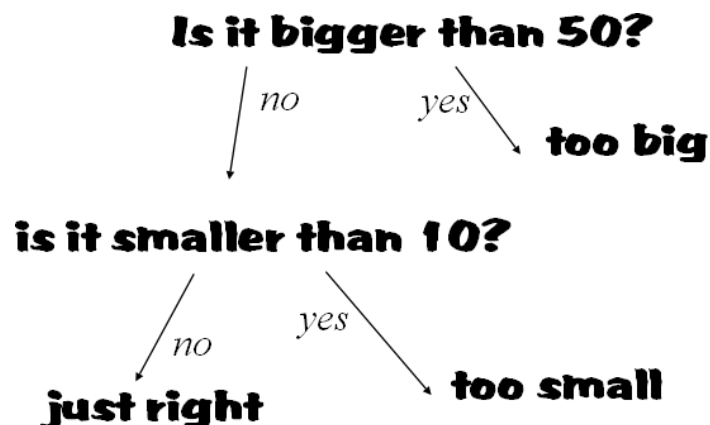
3. Copy out the number input program and get it working.
4. Write a program to have the user input a price. You should then check whether the price is less than 10. If it is, then print "You can afford it." Otherwise, print "It is too expensive."

Remember we can just use the variable name with no \$ at the end for numbers.

=	is equal to
<>	is not equal to
<	is less than
>	is more than
<=	is less than or equal to
>=	is more than or equal to

You can put one IF statement inside another, like this.

```
INPUT "What is your number: "; number
IF number>50 THEN
  PRINT "Too big"
ELSE
  IF number < 10 THEN
    PRINT "Too small"
  ELSE
    PRINT "Just right"
  ENDIF
ENDIF
```



Looping in BASIC –FOR loops

We can use a FOR loop to go through the same section of code more than once.

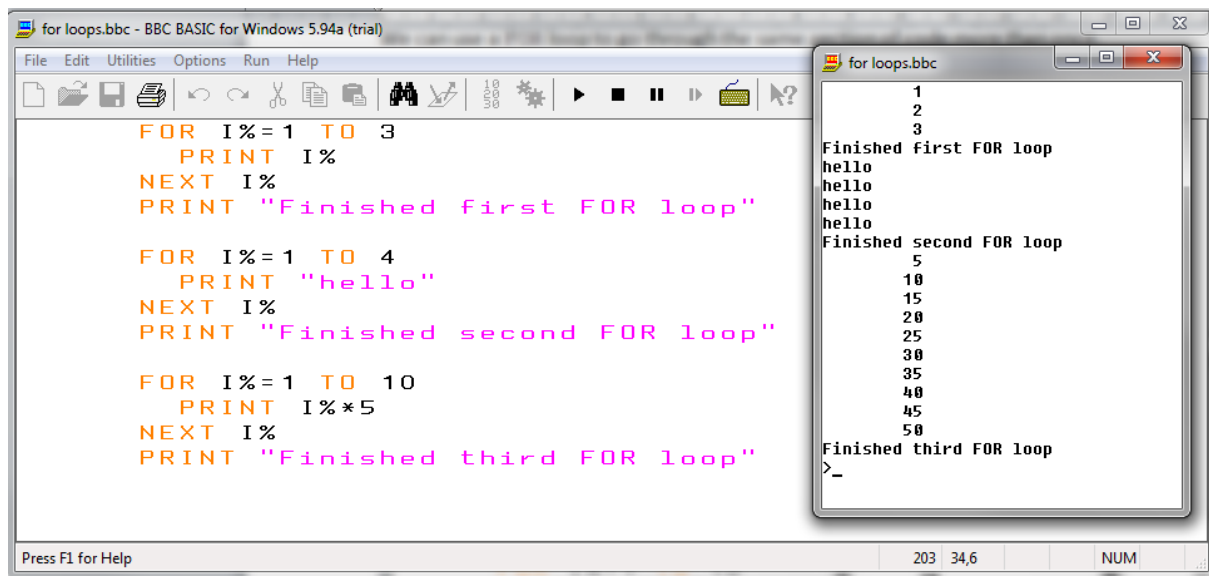
```
FOR I%=1 TO 4
    PRINT "hello"
NEXT I%
PRINT "Finished"
```

```
FOR I%=1 TO 10
    PRINT I%*5
NEXT I%
PRINT "Finished"
```

I will only contain the numbers 1, 2, 3 and then 4, so we can tell the program it will be an integer (whole number) by putting % at the end.

When it reaches the line NEXT I% it will add 1 to I and go back to the top of the loop.

If I is bigger than the top value (1-4) it will stop looping and continue below the loop instead.



The screenshot shows the BBC BASIC for Windows 5.94a (trial) interface. The main window displays the following code:

```
FOR I%=1 TO 3
    PRINT I%
NEXT I%
PRINT "Finished first FOR loop"

FOR I%=1 TO 4
    PRINT "hello"
NEXT I%
PRINT "Finished second FOR loop"

FOR I%=1 TO 10
    PRINT I%*5
NEXT I%
PRINT "Finished third FOR loop"
```

The output window shows the results of the program execution:

```
1
2
3
Finished first FOR loop
hello
hello
hello
hello
Finished second FOR loop
5
10
15
20
25
30
35
40
45
50
Finished third FOR loop
>_
```

Programming time 6!

Write your own loop to print the 6 times table. Your printout should look like this:

```
1x6= 6
2x6= 12
3x6= 18
```

and so on.

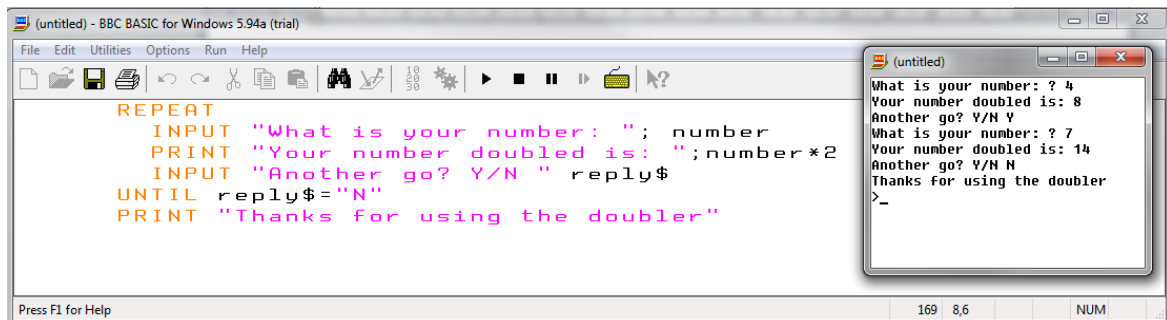
Looping in BASIC – REPEAT and WHILE loops

Sometimes we want to keep on doing something forever, or until something changes.

While it's raining, we'll play inside.

We'll keep playing that song until we get bored of it.

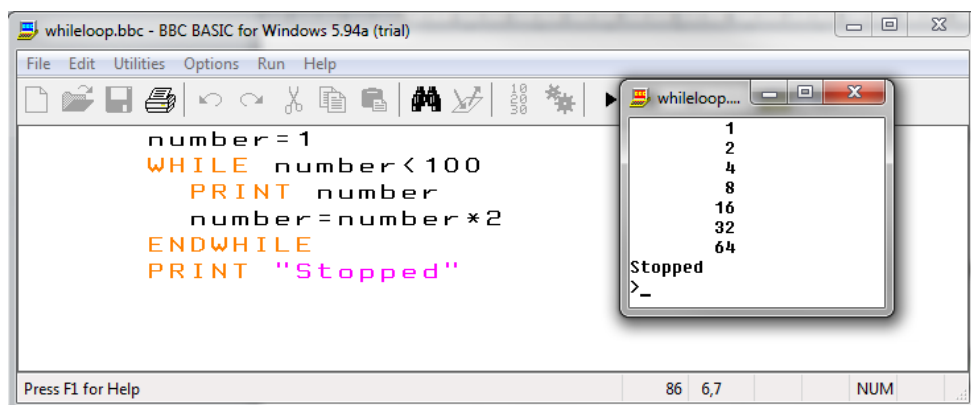
We can do this in BASIC too. One use for this is to run a program over and over until the user enters something to stop it. When the user enters anything other than N in response to Another go? Y/N the code within the REPEAT section will repeat. When N is entered the looping stops and the program continues.



```
REPEAT
  INPUT "What is your number: "; number
  PRINT "Your number doubled is: "; number*2
  INPUT "Another go? Y/N " reply$
UNTIL reply$="N"
PRINT "Thanks for using the doubler"
```

What is your number: ? 4
Your number doubled is: 8
Another go? Y/N Y
What is your number: ? 7
Your number doubled is: 14
Another go? Y/N N
Thanks for using the doubler
>_

A WHILE loop performs a test first. If the test is true, the code loops, but if it is false it continues after the loop.



```
number = 1
WHILE number < 100
  PRINT number
  number = number * 2
ENDWHILE
PRINT "Stopped"
```

1
2
4
8
16
32
64
Stopped
>_

REPEAT loops always run at least once. WHILE loops will not run at all if the test at the beginning is false.

Both will loop until the test is false – if the test is never false, it will loop forever!

Programming time 7!

1. Write a program to ask for a number and print out half the number, then ask if the user wants to continue.
2. Write a program to ask what number the user wants to start with. The program should multiply their number by 4, print it, multiply it by 4 again, print it again and keep going while the number is less than 1000. If they type in a number that is already bigger than 1000 the program will just print Finished instead of looping.