More Uses For FOR-NEXT Loops

FOR-NEXT loops are often used to create delays in the program. There are times when you don't want everything done at maximum speed and so you introduce a delay. You probably found that the answers in the MULTIPLY BY 10 program flashed up so quickly they seemed instantaneous. Let's make the computer look as if it's having to think before it answers by using FOR-NEXT to insert a delay. Add the lines shown in blue type to your program.

```
10 REM MULTIPLY BY 10
20 FOR X = 1 TO 8
30  PRINT "TYPE IN A NUMBER"
40 INPUT A
50 LET A = A * 10
52 FOR D = 1 TO 1000
54 NEXT D
60 PRINT "YOUR NUMBER MULTIPLIED BY 10
   IS ";
70 PRINT A
80 NEXT X
90 END
```

We have added another two lines, 52 and 54, inside our original FOR-NEXT loop. Let's look at them.

```
52 FOR D = 1 TO 1000
54 NEXT D
```

D is set to 1 and the program goes to the next line. This is the corresponding NEXT statement. Nothing actually happens inside the loop, the program simply loops back to line 52 and increments D to 2. This happens 1000 times before the program goes to the next part — which is printing the answer. Computers are fast, but everything takes a finite time, so looping back 1000 times takes a noticeable amount of time. Computers vary in the time they take to loop. On the Epson HX-20 this FOR-NEXT loop takes 2.9 seconds, while on the Spectrum it takes 4.5 seconds. Experiment by changing the number you use as the upper limit in line 52.

To make the computer behave more like a human being, add these three lines:

```
56 PRINT "NOW LET ME SEE..."
57 FOR E = 1 TO 1000
58 NEXT E
```

LIST the program and RUN it. We now have two delays that do absolutely nothing except waste time.

Add these two lines:

```
51 REM THIS LOOP WASTES TIME
55 REM THIS WASTES MORE TIME
```

Now LIST the program and have a good look at it. Notice how all the extra lines we have added have fitted into exactly the right places. Which brings us to the last point in this instalment of the course — line numbers.

We started our original program with line 10 and went up in jumps of 10 for each new line, ending with line 90. We could have chosen any numbers, for example 1, 2, 3 . . . 9. But if we had done that, how would we have fitted in the extra lines? Programmers always have afterthoughts and improvements to make, so allow for these by leaving big gaps between line numbers in the 'Mark I' versions of their programs. You could even start with line number 100 and go up in jumps of 50 or 100 if you wanted.

Some versions of BASIC include a useful command called AUTO. BBC BASIC has it, so does the Epson HX-20. The Dragon, Sinclair computers and the VIC 20 do not. If your BASIC has AUTO you can save a lot of time by having the line numbers generated for you automatically. Find out if your BASIC has AUTO by typing:

```
AUTO 100, 10<CR>
```

If your BASIC does have AUTO you will see on the screen:

```
100
```

The screen shows the number 100 followed by a space and then the *cursor*. The cursor is a mark (sometimes a line, or a square) that shows on the screen where the next character will appear. You can start entering the first line of the program from the cursor position. When you hit <CR> the next line will appear automatically, starting with the line number 110. AUTO, if you have it, can either be used by itself, or with one or two 'arguments'. *Argument* is a mathematical term. In the expression 2 + 3 = 5, the arguments are 2 and 3. With the AUTO command, it can be used just by itself (i.e. AUTO<CR>) or with one 'argument' (e.g. AUTO 100<CR>) or with two arguments (e.g. AUTO 300,50). AUTO by itself usually causes line numbers to start with 10 and to go up in increments (jumps) of 10. If only one argument is used (e.g. AUTO 100<CR>) the first number will be 100 (in this case) and then the numbers will go up in the 'default value' — which again is usually 10. If you specify two arguments, the first number specifies the starting line number and the second number specifies the increment. AUTO 250,50<CR> will give a starting number of 250, the next number will be 300 and so on in increments of 50. Even on the simplest micro, you're unlikely ever to run out of lines.

In the next instalment of this course we will look at various ways of improving the visual presentation of the program on the screen and different ways of printing out data.

Basic Flavours

IF — Most microcomputers can use this instruction in the form of either IF A > 999 THEN 80 or IF A > 999 GOTO 80. (The Spectrum uses IF A> 999 THEN GOTO 80.)

AUTO — This command is not available on the Commodore VIC 20, DRAGON 32 or Sinclair Spectrum.