construct and 'procedures'. Before a procedure can be used it must first be defined, and we will see how this is done in the version of the program for the BBC below. Notice that BBC BASIC defines the direction of data flow in the 'open' statement, using either OPENOUT or OPENIN:

```
10 DIM C$(2,7)
20 FOR R = 1 TO 7
30   FOR C = 1 TO 2
40     READ C$(C,R)
50   NEXT C
60 NEXT R
70 DATA 1,13.6,2,9.6,3,11.4,4,10.6,5,11.5,6,11.1,7,10.9
80 INPUT "TYPE S TO SAVE DATA",K$
90 IF K$ <> "S" THEN GOTO 80
100 PROCSAVE: CLEAR: DIM C$ (2,7)
110 INPUT "REWIND DATA TAPE THEN TYPE L", K$
120 IF K$ <> "L" THEN GOTO 110
130 PROCLOAD
140 PRINT "DAY    TEMP"
150 FOR R = 1 TO 7
160   FOR C = 1 TO 2
170     PRINT C$(C,R);"          "
180   NEXT C: PRINT
190 NEXT R
200 END
300 DEF PROCSAVE
310 X = OPENOUT ("TEMPDAT")
320 FOR R = 1 TO 7
330   FOR C = 1 TO 2
340     PRINT # X, C$(C,R)
350   NEXT C
360 NEXT R
370 CLOSE #X
380 ENDPROC
400 DEF PROCLOAD
410 X = OPENIN ("TEMPDAT")
420 FOR R = 1 TO 7
430   FOR C = 1 TO 2
440     INPUT # X, C$ (C,R)
450   NEXT C
460 NEXT R
470 CLOSE # X
480 ENDPROC
```

One of the advantages of BBC BASIC is that the file handling statements work equally well for cassette files or disk files, and programs written to run on a cassette can be used with no modification if a disk drive is added later.

So far we have seen how data can be transferred from DATA statements via arrays to data files on tape (or disk) and vice versa. The next step will be to look further at the INITIALISATION process to see exactly how many arrays will be needed, how many elements each will need and at what points in the program data will need to be transferred into and out of them.

Exercise

Write a program with the following components:

```
BEGIN
  INITIALISE
  ENTER DATA
  CHOOSE
    Save data
    Load data
    Exit program
END
```

INITIALISE will initialise any variables and arrays needed by the program. The data will comprise, say, 15 names, entered from the keyboard with prompts on the screen. CHOOSE will provide the user with a menu asking if you want to SAVE DATA?, LOAD DATA? or EXIT PROGRAM? See if you can create a flag in the EXIT PROGRAM? part that will automatically save the data if, and only if, a SAVE has not already been carried out.

Basic Flavours

The BBC Micro supports long variable names such as FULLNAME$. The Spectrum allows long numeric variable names, but only single letter string variable names. The Dragon 32, Vic-20 and Commodore 64 support long variable names, but only the first two characters are significant. On the Oric-1 variable names must consist of two characters (a letter followed by a letter or numeral), while the Lynx requires single letter names.

On the Dragon 32 you must use this format:

```
OPEN"O",#—1, "TEMPDAT"
PRINT#—1,1,13.6,2,9.6,3,11.4, etc.
CLOSE#—1
```

and

```
OPEN"I"#—1,"TEMPDAT"
INPUT#—1,D,T
CLOSE#—1
```

On the Commodore 64 and the Vic-20 use this format:

```
OPEN 1,1,2,"TEMPDAT"
PRINT 1,1:PRINT 1,13.6:PRINT 1,2: etc.
CLOSE 1
```

and

```
OPEN 1,1,0,"TEMPDAT"
INPUT #1,D,T
CLOSE 1
```

The Lynx and the Oric-1, in their standard form, do not support cassette files. For the 96K Lynx, see Issue 24.

The ZX81 does not support READ and DATA structures, so use this program:

```
10 DIM C$(14,4)
20 FOR X = 1 TO 14
30 PRINT "INPUT ENTRY NO.";X
40 INPUT C$(X)
50 NEXT X
60 STOP
```

Add lines 90-120 from the Spectrum program. When execution stops at line 60, SAVE the program. When you LOAD it again, array C$ will contain the data. Do not type RUN, which would reset the variables to 0; instead,use GOTO 100

TONY LODGE