predictably disastrous consequences for its surroundings.

As these development stages are completed, so familiarity and satisfaction combine to convince the programmer that the program works now, will always work and will never need changing, and anyway the code is a model of clarity. But programmers, not programs, need documentation. No program is self-explanatory, and there are always reasons for wanting to change working programs. Like any other mechanism, they need *maintenance*, and maintenance means manuals. Programs should be internally documented (using REM lines) for the programmer's benefit, and externally documented with accompanying literature for the sake of the user — even if the user is the programmer.

All of these lessons once had to be learned by mainframe programmers, and have been ignored and painfully rediscovered by microcomputer programmers. Taken together they comprise a programming 'structure', a unified approach to problem-solving far more comprehensive than a book of cautionary tales about avoiding GOTOs or embracing WHILE...WEND. Efficient programs are written by efficient programmers on a basis of structured experience and logical thinking. This series of articles aims to encourage both.

IAN McKINNELL

Bar Charts
The colours and depths of the bars forming the chart are easily adjusted in the program by changing the values of the control variables

```
399 REM**************************
400 REM*   3-D BAR CHARTS       *
401 REM**************************
500 GOSUB 1500:          REM INITIALISE
720 YY=22:XX=2:          REM ORIGIN COORDS
760 GOSUB 3200:          REM DRAW AXES
800 FOR E=LT-1 TO 0 STEP -1
820 OC=XX+E*DB:OL=YY-E*DB
840 GOSUB 2200:          REM INPUT DATA
900 FOR D=1 TO NN
920 HT=DT(D)
940 X0=OC+(D-1)*(BB+LT*DB):Y0=OL-HT-DB
960 GOSUB 4000:          REM PRINT BAR
980 NEXT D
1000 NEXT E
1100 XP=10:YP=23:GOSUB 3500
1120 PRINT"THREE-DIMENSIONAL HISTOGRAM"
1200 A$=INKEY$:IF A$="" THEN GOTO 1200
1400 END
1499 REM**************************
1500 REM*  INITIALISE  S/R      *
1501 REM**************************
1520 CL$=CHR$(147):   REM CLEAR SCREEN
1540 PRINT CL$
1560 PO$=CHR$(19):    REM HOME CRSR
1580 RT$=CHR$(13):    REM <RETURN>
1600 BB=2:DB=1:       REM BAR DIMENSIONS
1620 SW=40:SD=25:     REM SCREEN DIM'S
1640 HB=SD-DB:        REM MAX BAR HEIGHT
1660 DIM B$(HB+DB)
1680 FOR K=1 TO SD:PO$=PO$+RT$:NEXT K
1800 DIM DT(SW)
1900 GOSUB 2400:      REM BUILD BAR
2100 LT=4:            REM DEPTH FACTOR
2190 RETURN
2199 REM**************************
2200 REM* INPUT DATA ARRAY S/R  *
2201 REM**************************
2220 READ NN
2240 FOR Z=1 TO NN:READ DT(Z):NEXT Z
2310 DATA 6,12,10,4,7,8,10
2320 DATA 5,7 ,8 ,8,6,7
2330 DATA 6,7 ,4 ,8,5,3,9
2340 DATA 5,11,6 ,4,11,6
2390 RETURN
2399 REM**************************
2400 REM* BUILD WHOLE BAR S/R   *
2401 REM**************************
2500 TC$=CHR$(158):   REM SIDES=YELLOW
2520 FC$=CHR$(31):    REM FRONT=BLUE
2540 RV$=CHR$(18):    REM REVERSE ON
2560 NL$=CHR$(146):   REM REVERSE OFF
2580 CR$=CHR$(29):    REM CURSOR RIGHT
2600 CH$=CHR$(32):    REM SPACE CHAR.
2620 C1$=CHR$(169):   REM "◤" CHAR.
2640 C2$=RV$+C1$:     REM "◢" CHAR.
2680 FOR K=1 TO SW
2700 SP$=SP$+CH$
2720 RC$=RC$+CR$
2740 FF$=FF$+CH$
2760 NEXT K
2800 TL$=SP$+"/"
2820 BL$=SP$+NL$+C1$
2840 FL$=LEFT$(FF$,BB)
2900 L$=TC$+C2$+RV$+RIGHT$(TL$,BB)
2920 FOR K=1 TO DB
2940 B$(K)=LEFT$(RC$,DB-K)+L$+LEFT$(SP$,K-1)
2960 NEXT K
3000 L$=FC$+RV$+FL$+TC$+RIGHT$(TL$,DB)
3020 FOR K=DB+1 TO HB
3040 B$(K)=L$
3060 NEXT K
3100 L$=FC$+RV$+FL$+TC$
3120 FOR K=1 TO DB
3140 B$(HB+K)=L$+RIGHT$(BL$,DB+2-K)
3160 NEXT K
3190 RETURN
3199 REM**************************
3200 REM*    DRAW   AXES    S/R  *
3201 REM**************************
3300 PRINT TC$:           REM SET COLOUR
3320 FOR Y=2 TO YY-1
3340 XP=XX-1:YP=Y:GOSUB 3500:PRINT"!"
3360 XP=XX+YY-Y:GOSUB 3500:PRINT"/"
3380 NEXT Y
3400 YP=YY
3420 FOR X=XX-1 TO SW-1
3440 XP=X:GOSUB 3500:PRINT"-"
3460 NEXT X
3490 RETURN
3499 REM**************************
3500 REM* PUT CRSR  @ XP,YP S/R  *
3501 REM**************************
3600 PRINT LEFT$(PO$,YP)TAB(XP-1);
3620 RETURN
3999 REM**************************
4000 REM* PRINT PARTIAL BAR S/R  *
4001 REM**************************
4100 FOR V=1 TO HT
4120 XP=X0:YP=Y0+V-1
4140 GOSUB 3500:      REM PLACE CRSR.
4160 PRINT B$(V)
4180 NEXT V
4200 FOR V=1 TO DB
4220 XP=X0:YP=Y0+HT+V-1:GOSUB 3500
4240 PRINT B$(HB+V)
4260 NEXT V
4490 RETURN
READY.
```

Structure
Modular, and reasonably self-explanatory

Documentation
This routine is obviously crucial, but is entirely unexplained

Variable Names
Well commented, but not very meaningful

Completeness/Correctness
Does this work for all values? Does it produce the right output in all cases — when a value is less than zero, for example?

Error Trapping
No checksum, no scaling, nor error handling

Documentation
Good: no 'magic numbers', no mysterious control characters, everything translatable

Curate's Egg

This program to display three-dimensional bar charts is an annoying mixture of good and bad style: the internal documentation is good where it exists and the structure is modular, but the program is not self-explanatory, there is no error-trapping, and no user documentation

Basic Flavours

This program is written in Microsoft BASIC, and should run unchanged on micros with a 40×25 screen display. The screen dimensions are initialised in line 1620. The control character values initialised in subroutines 1500 and 2400 are for the Commodore 64; consult the ASCII chart in your manual for other machines