

THE AMSTRAD USER

Issue No. 9

October 1985

CONTENTS

Editorial.....	2
Letters.....	3
User Group Information.....	6
Utility Trio from Brenton Ross.....	8
The Trials of Tony Blakemore.....	13
The Learning Centre - Introduction to Music.....	14
Preliminary result of Competition.....	17
A Case for 'Arnold'.....	18
Junior Jotters.....	19
Bughunt - a game from Andre Urankar.....	21
Calendar Generator.....	23
Book Review.....	24
Hints from SA and ACT.....	25
Super-Quiz from Carl Allen.....	26
Sorting Methods - Part Two.....	29

For Tape Subscribers, the programs/routines can be found at these approximate counter readings:

Side 1 - RSXGEN:3, SCNDMP:24, MUSLIST1:39, JIBARCH:73, JIPICS:97, BUGHUNT:108, CALENDAR:138

Side 2 - QUIZ:3, SORTS:39

All enquiries and contacts concerning this Publication should be made to The Amstrad User, Shop 2, 33 The Centreway, Blackburn Road, Mt. Waverley, Victoria 3149, Australia. [Telephone: (03) 232 7055].

The Amstrad User is published each month by Strategy Publications. Reprinting of articles in The Amstrad User is strictly forbidden without written permission. All rights reserved. Copyright 1985 by Strategy Publications.

The single copy price of \$3.00 is the recommended retail price only. The subscription rate (for Australia only) is \$30.00 for 12 issues of the magazine only, or \$70.00 for 12 issues of the magazine plus tape containing all programs

appearing in that issue. Postage is included in the above prices. Overseas prices available upon application.

Please note that whilst every effort is made to ensure the accuracy of all features and listings herein, we cannot accept any liability whatsoever for any mistakes or misprints.

Contributions are welcome from readers or other interested parties. In most circumstances the following payments will apply to published material: Letters-\$5.00, Cartoons-\$5.00 and a rate of \$10.00 per page for programs, articles etc.

Contributions will not be returned unless specifically requested coupled with suitable stamped and addressed padded bag (for tapes) or envelope.

THE AMSTRAD USER

G'day,

A few months ago, July in fact, we reviewed the new CPC664. In the Editorial of that month I said that I would probably stick with my 464 until I needed something more powerful. Well, I didn't have to wait long before being tempted by Amstrad with the announcement of the new CPC6128. As its name suggests it has 64K more memory than the 464 or 664, is supposed to be software compatible with the 664 and runs CP/M 2.2.

Having hardly taken a breath, news broke of yet another addition to the range - this time the PCW8256. It is a self-contained word processing system with 256k of RAM, a built-in disc drive and a printer. A second drive of 1 megabyte can be added.

Amstrad are certainly heaping the pressure on their competitors, and our reviews of these two new machines next month will let you decide if their efforts are worthwhile.

The Competition closed on August 15th and, quite frankly, we were overwhelmed with the response. It has taken far longer than we had anticipated to sift through the entries, but at least we have got a result from the Education Class which you will find on Page 17. Apart from the sheer number of entries, our efforts were hampered in many cases by less than adequate documentation, programs failing to do what they were supposed to do, read errors on tapes and even syntax errors! With fingers crossed, we will do our level best to have the final results published next month.

Talking about next month, we plan to include another 8-page supplement, this time covering a host of new books which are becoming available to improve your skills on the Amstrads. Judging by my post, the supplement last month appears to have been welcomed (despite the printer transposing two pictures!) and we will include another in the December issue. There will also be a Competition for the opportunity to win ten books for your 'Amstrad Library'.

See you next month,

Ed

Letters

My husband is a surveyor and wishes to use the Amstrad in conjunction with his printer (NSD-WP1000) to obtain hardcopies of plotting. He is experiencing some difficulty in "cracking the code" of the printer for graphics printing. The code is: ESC K+n1+n2.

If you have any relevant information regarding obtaining hard copy prints of plotted points or could suggest a book or software concerning this it would be greatly appreciated. If not, perhaps this query could be printed in your magazine.

Margaret Hudson, Wentworth Falls
N.S.W.

The solution to sending control characters to the printer is to send them using the CHR\$(n) function. No doubt other users have found other methods, but this is the one that appears to be most consistently successful.

For example the particular sequence mentioned is ESC K + n1 + n2. This translates into:-

```
PRINT #8,CHR$(27);
CHR$(75);CHR$(n1);
CHR$(n2)
```

The trick to sending n1 and n2 correctly is to send the ASCII code for the number that you wish to send. Thus 0 is &30, 1 is &31, 2 is &32, etc. The equivalent decimal values are: 0 is 48, 1 is 49, 2 is 50, etc.

This should solve the primary difficulty. The next problem to overcome is to translate the graphics into the necessary bytes to produce the required graphics on the printer. BASIC is terribly slow at doing screen dumps. From the description of their printer, it

would suggest that it probably is not very suitable for graphics (unless it is a dot matrix printer - and the model number suggests that it is a daisy wheel). If it is a dot matrix, it will be necessary to determine the order in which the bits in a byte fire the pins and then work out the value of the byte which contains the bits set in the correct places to produce the "bit image" required. If you are trying to do a screen dump, this becomes more complicated still for several reasons, and this may be the subject of an article in a future issue.

C.W. Hall from Newcastle, enquires in the September issue about problems in hooking up a 5.25" disc drive to the Amstrad. He was apparently told by AWA that it cannot be done, unless costly modifications are carried out.

In actual fact, it is an extremely simple procedure, requiring no modifications at all to existing equipment.

The ribbon cable, supplied with the original DDI-1 drive, is already fitted with a piggy-back 34 pin female plug, placed there for the purpose of connecting a second Amstrad drive. All that is required now, is making up of another 34 wire cable, say 18" long, with a male connector on one end and a female connector at the other. However, great care must be taken to ensure that correct polarity is maintained. If you have no experience in that sort of thing, have the cable made up. If the 5.25" drive, which you intend to use, has no internal power supply, suitable voltages must be supplied externally.

It is worth noting however, that the 5.25" drive cannot be used on its own in the system. The DDI-1 drive must be switched on, as it supplies power to the drive interface (the grey box, plugged into the computer).

Henry Denver, Forest Hill, Vic.

Some four months ago my family purchased an Amstrad CPC 464. I was very pleased with the computer and especially Locomotive basic. After a couple of months with the computer I was getting tired of the same old writing that the computer prints out. So I devised a short routine that changes all the small letters (CHR\$(96) to CHR\$(122)) to computer style letters that appear on computers in TV shows such as the "WHIZZ KIDS". I have found that this enhances titles, game statistics etc. I hope this routine is useful to other readers of your User magazine.

10	SYMBOL AFTER 96
20	SYMBOL 97,60,36,36, 126,98, 98,98,0
30	SYMBOL 98,124,68,68, 126, 98,98,126,0
40	SYMBOL 99,126,66,64,96, 96,98,126,0
50	SYMBOL 100,126,66, 66,98, 98,98,126,0
60	SYMBOL 101,126,64, 64,126, 96,96,126,0
70	SYMBOL 102,126,64, 64,126, 96,96,96,0
80	SYMBOL 103,126,66, 64,102, 98,98,126,0
90	SYMBOL 104,66,66,66,126, 98,98,126,0

```

98,98,98,0
100 SYMBOL 105,16,16,16,24,
24,24,24,0
110 SYMBOL 106,4,4,4,6,70,
126,0
120 SYMBOL 107,68,68,68,126,
98,98,98,0
130 SYMBOL 108,64,64,64,96,
96,98,126,0
140 SYMBOL 109,254,146,146,
210,210,210,210,0
150 SYMBOL 110,126,66,66,98,
98,98,98,0
160 SYMBOL 111,126,70,70,66,
66,66,126,0
170 SYMBOL 112,126,66,66,126,
96,96,96,0
180 SYMBOL 113,126,66,66,66,
66,78,126,0
190 SYMBOL 114,124,68,68,126,
98,98,98,0
200 SYMBOL 115,126,66,64,126,
6,70,126,0
210 SYMBOL 116,126,16,16,24,
24,24,24,0
220 SYMBOL 117,66,66,66,98,
98,98,126,0
230 SYMBOL 118,98,98,98,102,
36,36,60,0
240 SYMBOL 119,146,146,146,
210,210,210,254,0
250 SYMBOL 120,66,66,66,60,
98,98,98,0
260 SYMBOL 121,66,66,66,126,
24,24,24,0
270 SYMBOL 122,126,66,2,126,
96,98,126,0

```

Rowland Hayes, Rainbow Flat, NSW.

I cannot understand how Mr. Brown gets the address of the variable into the register DE (Bytes and Pieces, August '85, p.25, routine #1). According to the manual, the passed parameters are stored somewhere in memory and the IX register points to the least-significant byte of the last one.

If we take CALL routine, number, @k as an example, the address of k will be stored at (ix+0) and (ix+1), and the value of number at (ix+2) and (ix+3), in both cases in the 1sb-msb order.

(number will be forced into integer format if not defined as integer).

I am enclosing a short routine, which flashes the cursor while waiting for a key press, then returns with the key code in the variable k. It could, of course, be done in BASIC by printing CHR\$(2) and CHR\$(3) to turn the cursor off and on, and INKEY\$ to get the character.

The three ROM calls preserve all registers except for &bb09, which uses the AF register to get the character in A and indicate in F whether a key was pressed.

```

160 CLS:PRINT"GLASS TYPEWRITER.
"
```

```

110 PRINT"LKS 850818."
120 PRINT"Waits for a key press."
130 PRINT"flashes cursor while
waiting."
140 PRINT"echoes key to screen."
150 PRINT"Try holding down CTRL
and"
160 PRINT"pressing keys '@' to ' '."
170 PRINT"Exit by large
ENTER."PRINT
180 MEMORY 30000/Anywhere
convenient
190 DEFINT f-v:DEFSTR w-z
200 DEF FNh=VAL("&" +z)' Convert
Hex
keyget=HIMEM+1:k=0
220 FOR i=keyget TO keyget+22
230 READ z:POKE i,FNh:NEXT i
240 L1:Turn off cursor:call &bb84
250 DATA cd,84,bb
260 'Delay loop:ld b,0;l2:djnz L2
270 DATA 06,00,10,fe
280 'Read key:call &bb09
290 DATA cd,09,bb
300 'Turn on cursor:call &bb81
310 DATA cd,81,bb
320 'Repeat if no key pressed:jr nc,L1
330 DATA 30,fi
340 'Get address of k into hl reg.:
350 lsb:ld l,(ix+0)
360 DATA dd,6e,00
370 'msb:ld h,(ix+1)
380 DATA dd,66,01
390 'load key character into lsb of k:
400 ld (hl),a
410 DATA 77
420 'Return
430 DATA c9
440 PRINT"Enter delay (0 to 255)"
450 INPUT"0=slowest,1=faster,"z:
z=LEFT$(z,3)
460 PRINT:POKE keyget+4,FNh AND

```

255

```

470 WHILE k<>13 OR INKEY(18)
480 CALL keyget,@k:PRINT CHR$(1)
CHR$(k);
490 WEND:PRINT:PRINT
500 STOP

```

P. Lukes, Toowoomba, Qld

I have a tip directed to those who have bought AMSWORD/TASWORD 464 Version 1.02 and wish to use TASPRINT 464 with it.

The TASPRINT manual is not very clear on how to achieve output with this combination. Section 5 of this manual gives the method for modifying AMSWORD/TASWORD but does not tell you what to do with the modified version to allow loading and using TASPRINT with it. The answer is stupidly simple. It appears that "TP" modifies Version 1.02 to be equivalent to later versions and then you simply follow the instructions in Section 5.2.

On the back of the TASPRINT 464 manual, there is a list of supported printers. The program itself lists an extra one which opens up a whole tin of worms. The extra printer listed is the NEC PC-8023B-N. This is only one name under which this printer was marketed. From an American magazine I have found out that this printer was marketed, at least there, as an ADS 8001, PMC DMP85 and the C-Itoh 8501. The printer is manufactured by TEC in Japan and was marketed under that name also. I have found out that one of the printers that Dick Smith marketed under a proprietary name in the 'early' days (1979 and early 1980's) is in fact a detuned version of the C-Itoh. Thus the fonts for the NEC printer will work with all of these printers.

I have a problem that someone out there may have solved or someone may be able to solve. The pinout diagram on Appendix V Page 2 shows that the printer port carries signals for 8 data bits. Why does the printer respond only

to 7 data bits and how can the 8th bit be sent? My C-Itoh has some wonderful additional features that require use of the 8th bit to give ASCII characters above 127 and I would like to be able to use them. Can anyone help, please?

Arthur Harris, East Burwood, Vic.

SON of ARNOLD

*There was movement at the
C.A.U.S.*

*For the word had passed around
That the Son of "ARNOLD"
Had finally come to town.*

*Some had already seen him
Some had even had a go
But for most they'd have to wait
Till they saw him at the P.C.
Show.*

*There was much speculation
As to what the "Lad" could or
couldn't do
Some said he could jump through
hoops
But most pooh, poohed.*

*An Evil One amongst them
Even dumped young Arnold's
Dad
He'd gone by way of Trading
Post
To make way for the Lad.*

*But the Evil One had suffered
As he should, the cad
From "Keyboard Withdrawal"
While waiting for the Lad.*

*Now the King collects his taxes
And rubs his hands with glee
As he dreams of all the extra
software
That will come across the sea.*

*The poverty stricken peasants
Can only stand by in dismay
For they know that the
"Grandson of Arnold"
Is not far away.*

Don Leith, Brunswick, Vic

AMSTRAD ACHIEVERS

Get your name in our "HALL OF FAME"

In the next month or so we will publish proven high scores for games or adventures which have been achieved on an Amstrad computer. Register your name and score on the form below, and, if possible, send a photograph to put doubt out of everyone's mind!

Name.....

Address.....

Telephone Number.....

Game..... Score.....

Achieved (date)Game lasted (mins.secs).....

Signed.....

THIS NEXT PART MUST BE COMPLETED

Witness' Name

Address

Telephone Number

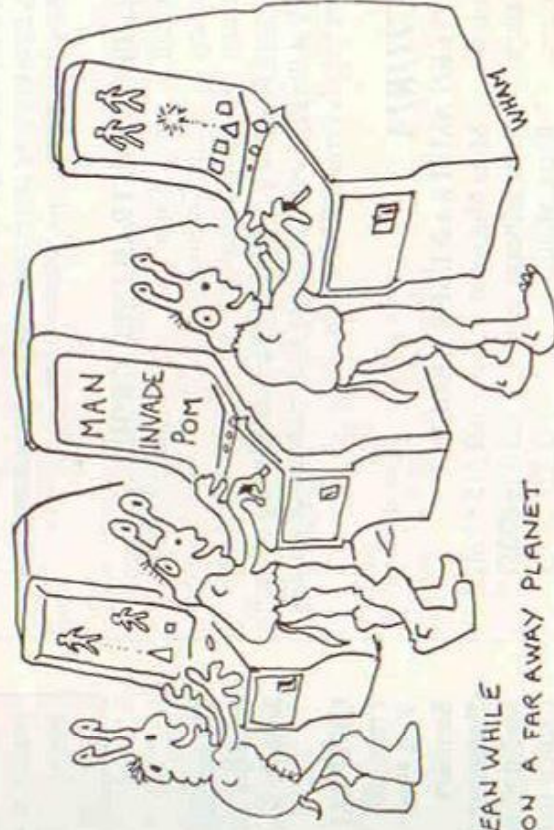
Occupation

I confirm that the above claimed score is accurate and genuine

Signed

Post this form along with your tips for playing the game to:

Amstrad Achievers, The Amstrad User, Shop 2, 33 The Centreway,
Blackburn Road, Mt. Waverley, Victoria 3149.



MEAN WHILE
ON A FAR AWAY PLANET

Utility Trio

from Brenton Ross

After noting the considerable interest which my first attempt at a screen dump aroused at the Canberra Amstrad User Group, I set about producing a more useable version - one that was independent of the mode setting and which was relocatable. During the process, I wrote several programs to relocate machine code and to convert them to a BASIC format. These were eventually combined into one program: RSXGEN. My final versions are presented below.

A machine code program to copy the screen to a printer.

Listing 1 is provided for those who have the DEVPAK Assembler or for those who wish to tailor the program to their specific printer. To cater for alternative printers, a few surplus zero bytes have been included in MSG1, MSG2 and MSG4. This should allow alternative escape sequences to be inserted. The changes can either be made to Listing 1 before re-assembling the code, or they can be made to Listing 2 before it is saved.

The escape sequences are for a Brother M-1009 printer and I believe they do work on others, however, it would be wise to check your manual first.

The first section of code, down to START: is only required if the code is implemented as a Resident System Extension (RSX). If you are incorporating it into a larger machine code program, you should leave it out.

A BASIC program to generate a screen dump RSX.

The program in Listing 2, when run, will add a new instruction !SCRDMP to the AMSTRAD then delete itself - so save it before you run it!

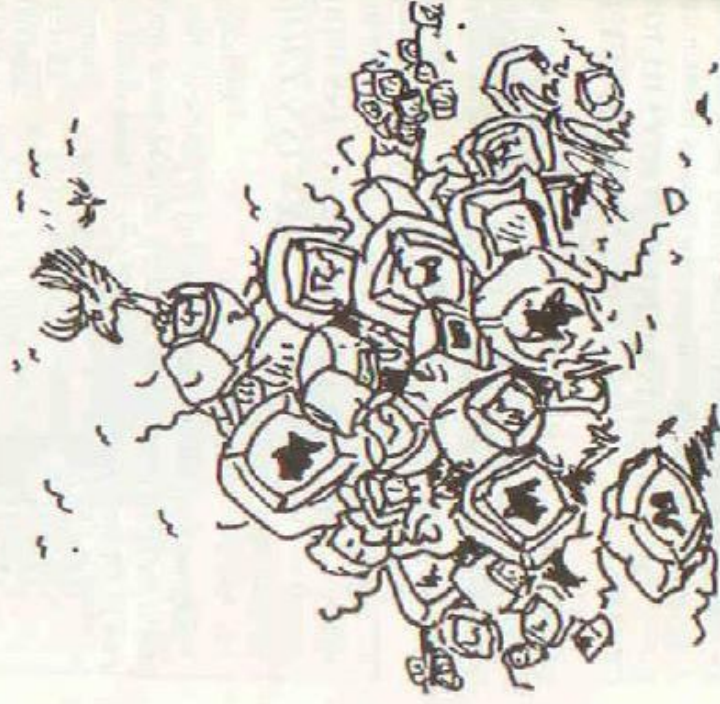
The program first reserves space at the top of available memory then pokes the code into that space. Next, it makes corrections to compensate for being at a location other than where it was originally assembled. Finally the first section is called which links the instructions into the operating system. It should be noted that:

1. The area displayed is the first quadrant with respect to the graphics origin. Therefore, to get the whole screen, first perform ORIGIN 0,0.
2. To abort printing, press the ESC key, however the key is only tested at the end of each line, so it may need to be held for a few seconds.

RSXGEN.BAS - a program to convert machine code to a Basic program.

The program in Listing 2 was automatically generated by the program in Listing 3. To use this program, it is first necessary to assemble the machine code *twice*, noting the start and end points. Obviously the two versions should not overlap, but also should *not* be separated by multiples of 256 and should be high in memory. Then exit from the Assembler, set MEMORY to just below the code, then load and run RSXGEN.BAS. You now have a program which will generate a new RSX, and is in a form suitable for publishing. So, if you have an interesting extension to Arnold, let's see it in print!

(Note: PRINT#9 has been used extensively in this program. Whilst there have been no reported errors using this command in similar circumstances, it would be prudent to use WRITE#9 instead - Ed)



Screen Dump?

Pass 1 errors: 00

```

10 ;*****
20 ;
30 ; SCREEN DUMP ROUTINE
40 ;
50 ; FOR AMSTRAD CPC464
60 ; AND BROTHER M-1009
70 ;
80 ;*****
90
0000 100 OFF: EQU 0
0001 110 ON: EQU 1
120
130 *D+
140
150 RSX:
160 IF RSX
170 ;RSX INTRODUCTION
180 LD HL,$
190 LD BC,COMAND
200 CALL #BCD1
210 RET
220 COMAND: DEFW NAME
230 JP START
240 NAME: DEFW "SCROM"
250 DEFB "P"+#80,0
260 END
270
280 START: ENT $
290 LD IX,ZZ
300 LD IX,MSG1
310 CALL PRINTER
320 LD BC,400
330 L41: LD A,C
340 SUB 4
350 LD C,A
360 LD A,B
370 SBC A,0
380 LD B,A
390 LD IX,MSG2
400 CALL PRINTER
410 LD DE,0
420 L61: CALL COLUMN
430 INC DE
440 LD A,#02
450 CP D
460 JR NZ,L61
470 LD A,#7F
480 CP E
490 JR NZ,L61
500 LD IX,MSG3
510 CALL PRINTER
520 PUSH BC
530 LD A,66
540 CALL #BB1E
550 POP BC
560 RET NZ
;CHANGE THIS TO 38000 TO GET SECOND
; VERSION FOR RSXGEN
;ONLY ASSEMBLE IF ITS TO BE RSX
;THESE COMMANDS GET OVERWRITTEN BY O/S
;POINTS TO JUMP TABLE
;KL LOG EXT
;A JUMP TABLE OF ONE
;THE NAME
;ACTUAL START DURING USE
; THE COLUMN OF DATA IS BUILT HERE
;SET LINE FEED RATE TO 1/18"
;400 "GRAPHICS" POINTS HIGH
;DOWN 2 LINES OF PIXELS
;SET BIT IMAGE MODE
;START AT LEFT EDGE
;SEND A COLUMN OF TWO PIXELS TO PRINTER
;ONTO THE NEXT COLUMN
;HAVE WE FINISHED THIS LINE?
;NO, SO ROUND AGAIN
;YES, SO LINE FEED THE PRINTER
;TEST ESCAPE KEY
;KM TEST KEY

```

```

37083 3E00      LD  A,0
37085 B8       CP  B
37086 20C0     JR  NZ,L41
37088 B9       CP  C
37089 20C9     JR  NZ,L41
37091 DD210791 LD  IX,MSG4
37095 CDEB90   CALL PRINTER
37098 C9       RET
37099          PRINTER:
37099 DD7E00   LD  A,(IX+0)
37102 FE00   CP  0
37104 C6      RET  Z
37105 CD2BBD  CALL #BD2B
37108 30F5    JR  NC,PRINTER
37110 DD25    INC  IX
37112 18F1    JR  PRINTER
37113          PRINTER:
37114 1B330C00 740 MSG1:  DEFB 27,"3",12,0,0,0
37120 1B2A047F 750 MSG2:  DEFB 27,"*",4,127,2,0
37126 0D0A00 760 MSG3:  DEFB 13,10,0
37129 1B400000 770 MSG4:  DEFB 27,"e",0,0,0,0
37135          COLUMN:
37135 C5       PUSH BC
37136 D5       PUSH DE
37137 60      LD   H,B
37138 69      LD   L,C
37139 E5      PUSH HL
37140 CDF0BB  CALL #BBF0
37143 FE00   CP  0
37145 2808    JR  Z,COL2
37147 FDCB20C6 SET  0,(Y+0)
37151 FDCB20CE SET  1,(Y+0)
37155 E1      POP  HL
37156 D1      POP  DE
37157 D5      PUSH DE
37158 23     INC  HL
37159 23     INC  HL
37160 CDF0BB  CALL #BBF0
37163 FE00   CP  0
37165 2808    JR  Z,COL3
37167 FDCB00D6 SET  2,(Y+0)
37171 FDCB00DE SET  3,(Y+0)
37175 FD7E00  LD   A,(Y+0)
37178 CD2BBD  CALL #BD2B
37181 30F8    JR  NC,COL3
37183 FD360000 LD  (Y+0),0
37187 D1      POP  DE
37188 C1      POP  BC
37189 C9      RET
37190 00      DEF 00
37190 00      DEF 00
37190 ZZ:     DEF 00

```

Pass 2 errors: 00

Table used: 199 from 436

LISTING 1

```

10 READ n: cs = n
20 s = HIMEM - n: MEMORY s - 1
30 FOR i = 1 TO n
40 READ a: cs = cs + a: POKE s
   +i-1,a
50 NEXT i
60 READ n: cs = cs + n
70 FOR i = 1 TO n
80 READ a,b: cs = cs + a + b
90 a = a + s: ah = INT(a/256):
   al = a - ah*256
100 POKE s+b-1,al: POKE s+b,ah
110 NEXT i
120 READ check: IF check = cs THEN
   N CALL s: NEW ELSE PRINT"ERRO
   R IN DATA STATEMENTS!"
130 END
140 DATA 191
150 DATA 33, 136, 144, 1, 14
   6, 144, 205, 209, 188, 2
   01
160 DATA 151, 144, 195, 158,
   144, 83, 67, 82, 68, 77
170 DATA 208, 0, 253, 33, 70
   , 145, 221, 33, 250, 144
180 DATA 205, 235, 144, 1, 1
   44, 1, 121, 214, 4, 79
190 DATA 120, 222, 0, 71, 22
   1, 33, 0, 145, 205, 235
200 DATA 144, 17, 0, 0, 205
   , 15, 145, 19, 62, 2
210 DATA 186, 32, 247, 62, 1
   27, 187, 32, 242, 221, 3
   3
220 DATA 6, 145, 205, 235, 1
   44, 197, 62, 66, 205, 30
230 DATA 187, 193, 192, 62,
   0, 184, 32, 204, 185, 32
240 DATA 201, 221, 33, 9, 14
   5, 205, 235, 144, 201, 2
   21
250 DATA 126, 0, 254, 0, 200
   , 205, 43, 189, 48, 245
260 DATA 221, 35, 24, 241, 2
   7, 51, 12, 0, 0, 0
270 DATA 27, 42, 4, 127, 2,
   0, 13, 10, 0, 27
280 DATA 64, 0, 0, 0, 0, 19
   7, 213, 96, 105, 229
290 DATA 205, 240, 187, 254,
   0, 40, 8, 253, 203, 0

```

```

300 DATA 198, 253, 203, 0, 2
   06, 225, 209, 213, 35, 3
   5
310 DATA 205, 240, 187, 254,
   0, 40, 8, 253, 203, 0
320 DATA 214, 253, 203, 0, 2
   22, 253, 126, 0, 205, 43
330 DATA 189, 48, 248, 253,
   54, 0, 0, 209, 193, 201
340 DATA 0
350 DATA 14
360 DATA 0, 2, 10, 5, 15, 1
   1, 22, 14, 190, 25, 114
   , 29
370 DATA 99, 32, 120, 47, 99
   , 50, 135, 56, 126, 71,
   99, 74
380 DATA 129, 94, 99, 97, 24
   909

```

```

10 'RSXGEN
20 ' This program takes two copi
   es of a m/c code program and
   generates
30 ' a BASIC program which when
   run will produce an RSX at t
   he top of memory
40 '
50 'Note: The two m/c code copie
   s should not overlap and must
   be separated by
60 'more than 256, but NOT a mul
   tiple of 256.
70 '
80 INPUT"Start Address of Versio
   n 1";sa1
90 INPUT"End Address of Versio
   n 1";ea1
100 INPUT"Start Address of Versio
   n 2";sa2
110 INPUT"File Name for the Gener
   ated Program";files$
120 OPENOUT files$
130 ln = 10
140 PRINT "Enter any comments tha
   t you want at the beginning"
150 PRINT"Terminate with just 'EN
   TER'"
160 LINE INPUT c$
170 WHILE c$ <> ""
180 c$ = STR$(ln)+""+c$

```

```

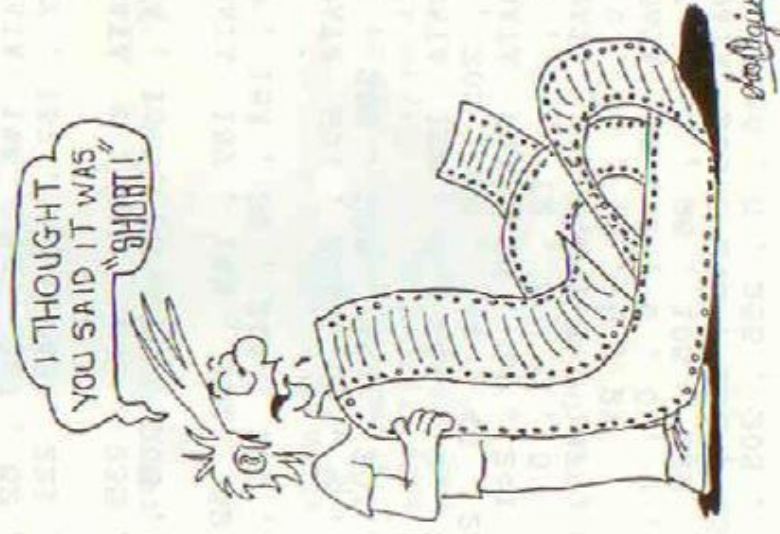
190 PRINT #9,cs
200 ln = ln + 10
210 LINE INPUT cs
220 WEND
230 PRINT #9,ln;"READ n: cs = n":
ln = ln + 10
240 PRINT #9,ln;"s = HIMEM - n: M
EMORY s - 1":ln = ln + 10
250 PRINT #9,ln;"FOR i = 1 TO n":
ln = ln + 10
260 PRINT #9,ln;" READ a: cs = c
s + a: POKE s+1-1,a":ln = ln
+ 10
270 PRINT #9,ln;"NEXT i":ln = ln
+ 10
280 PRINT #9,ln;"READ n: cs = cs
+ n":ln = ln + 10
290 PRINT #9,ln;"FOR i = 1 TO n":
ln = ln + 10
300 PRINT #9,ln;" READ a,b: cs =
cs + a + b":ln = ln + 10
310 PRINT #9,ln;" a = a + s: ah
= INT(a/256): al = a - ah*256
":ln = ln + 10
320 PRINT #9,ln;" POKE s+b-1,al:
POKE s+b,ah":ln = ln + 10
330 PRINT #9,ln;"NEXT i":ln = ln
+ 10
340 PRINT #9,ln;"READ check: IF c
heck = cs THEN CALL s: NEW EL
SE PRINT":
350 PRINT #9,CHR$(34);"ERROR IN D
ATA STATEMENTS!";CHR$(34):ln
= ln + 10
360 PRINT #9,ln;"END":ln = ln + 1
0
370
380 ' now generate the data state
ments for the base program
390
400 length = ea1 - sa1 + 1: cs =
length
410 PRINT #9,ln;"DATA ";length:ln
= ln + 10
420 s = sa2 - sa1: oi = 0
430 i = sa1: WHILE i <= ea1
440 j = 1: PRINT #9,ln;"DATA ";
450 WHILE i <= ea1 AND j <= 10
460 a = PEEK(i): cs = cs + a
470 IF a <> PEEK(i+s) AND i >
oi THEN oi = i + 1: n = n +
1
480 PRINT #9,a:
490 IF j < 10 AND i < ea1 THE
N PRINT #9,","; ELSE PRINT #9

```

```

500 i = i + 1: j = j + 1
510 WEND
520 ln = ln + 10
530 WEND
540
550 ' finally the correction data
for relocating
560
570 cs = cs + n
580 PRINT #9,ln;"DATA ";n:ln = ln
+ 10
590 i = sa1: WHILE i <= ea1
600 j = 0: line.flag = -1
610 WHILE i <= ea1 AND j <= 5
620 al = PEEK(i)
630 IF al = PEEK(i+s) GOTO 69
0
640 IF line.flag THEN PRINT #
9,ln;"DATA ";:line.flag = 0
650 i = i + 1: ah = PEEK(i)
: a = ah * 256 + al - sa1
660 PRINT #9,a,"";i-sa1;;
cs = cs + a + i - sa1
670 IF j < 5 THEN PRINT #9,
","; ELSE PRINT #9
680 j = j + 1
690 i = i + 1: WEND
700 ln = ln + 10
710 WEND
720 IF line.flag THEN PRINT #9,ln
;"DATA ";cs ELSE PRINT #9,cs
730 CLOSEOUT
740 END

```



The Trials of Tony Blakemore

A Column for the Absolute beginner

Animation is the art of tricking your eyes. Because of the after image that remains on the retina we can fool our eyes into thinking that images printed, erased and quickly printed again, slightly offset to the original, form continuous movement.

With the Amstrad we can move an object across the screen in many different ways. Last month with the 'Cars' program we moved an eight by eight grid, eight pixels at a time. This creates a sense of motion and although a little jerky passes for animation.

To make the animation more realistic, this month we are going to use two images instead of one. As the character is moved we will change it slightly.

Using the same program with a few alterations we will have six beetles racing. To animate the beetles we will make the legs move and the mouth open and close. Because of Melbourne's inclement weather you will notice the

beetles occasionally slipping on the wet track!

The two different grids are stored alternatively in b\$. As the program loops b\$ is changed between chr\$(201) and chr\$(202). This is done in line 84 where the initial value is set to 202; line 292 where b\$ is set to either chr\$(201) or chr\$(202); line 342 where the value is altered on each loop and line 342 where the value of b\$ is set to 201 if the previous run was 202.

For the new program load the CARS program from last month's issue and add the following lines:

```
30 'Loops and makes
    the beetles go
    faster
```

```
40 'Define
```

```
chr$(201), (202) as
a beetle
```

```
80 SYMBOL 201,195,
90,126,255,255,126,
90,195
```

```
82 SYMBOL 202,102,
90,126,253,253,
126,90,102
```

```
84 b=202
```

```
110 'Set start
location of the
beetles
```

```
220 LOCATE 10,14
```

```
230 PEN 3:PRINT
```

```
"AMSTRAD BEETLE
```

```
RACES':PEN 1
```

```
250 'Print beetle at
random location
```

```
260 'Check for first
```

```
beetle finished
```

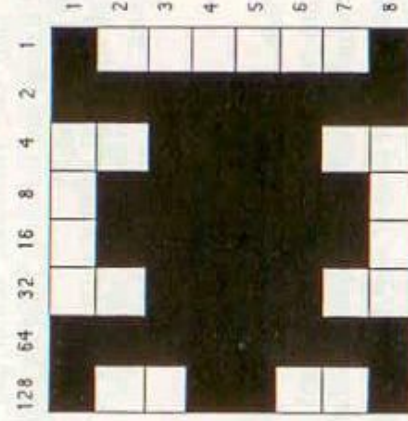
```
292 b$=CHR$(b)
```

```
300 PRINT " "+b$
```

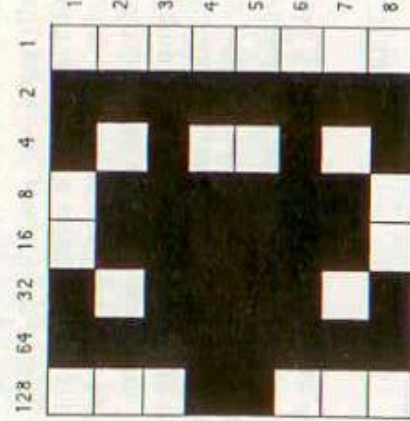
```
320 beetle=
```

```
INT(RND*2+1)
```

```
330 IF beetle=1 THEN
X(NUMBER) =
X(NUMBER)+1
342 IF b=201 THEN
b=202:GOTO 350
344 b=201
```



CHR\$(201)



CHR\$(202)

Now save the new program. The sense of animation is now much better. Any number of different characters can be used to create better animation. The more that are used the better the effect. Next month we will look at multi-coloured characters and how they can be grouped together to form larger characters.

TO OPEN



Turn to Page 18

The Learning Centre

An Introduction to Music - Part One

from Peter Campbell

Soundly Different

Most people can recognise sounds and can distinguish between them, but what makes one sound different from another? Each sound is made by a transfer of energy from the source, through whatever medium is available, to the receiver. Usually the medium will be air and the receiver will be your ear. However, sound can travel through other substances and the more solid they are the faster the sound will be transmitted.

The difference between sounds is attributable to three factors:

1. pitch;
2. volume; and
3. quality.

Pitch determines whether the sound is deep like a thunderclap or high like the squeak of a mouse. Volume determines whether the sound is deafening or so soft that you have to strain to hear it at all. Quality, however, is more elusive.

If you hear a clarinet play a particular note, you will not mistake it for a violin playing the same note. If you have a really good 'ear' you will be able to distinguish between a Stradivarius and a lesser violin. This attribute of sound is quality and can be explained best by looking at a graphical depiction of the sound, its waveform.

Figure 1 shows a 'pure' sound with a simple waveform. The distance between the 'peaks' of the graph is the wavelength, the number of times per second that the sound wave completes an oscillation is its frequency and the size of the peaks is a measure of its volume (or amplitude).

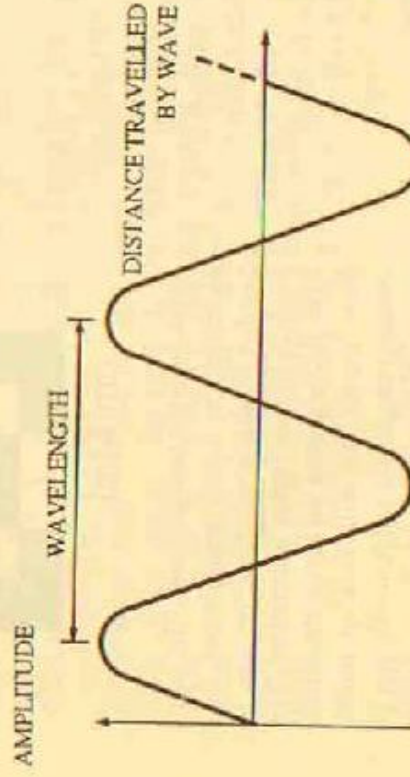


FIGURE 1

If some irregularities are added to the waveform, perhaps producing something like Figure 2, then the sound will have a different quality caused by the added 'noise'. Minor variations in pitch or volume will also produce different sounds and we will look at some of the possibilities later in the series.

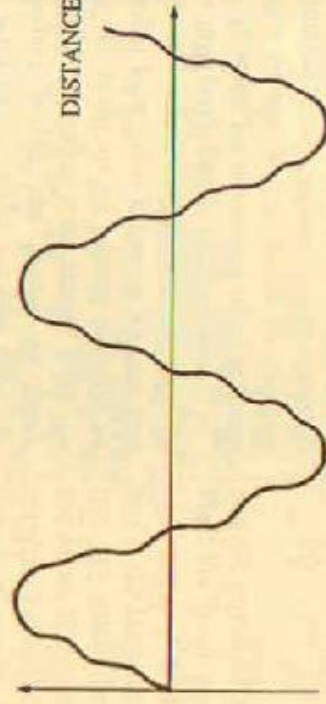


FIGURE 2

It's All Done With Numbers

It is said that the ancient Greek mathematician, Pythagoras (of right-angled triangle fame), weighed a group of blacksmiths' hammers to find out why they seemed to be playing in tune as the anvil was struck. He found that notes an octave apart were produced by hammers whose mass differed by a factor of two. That is, one hammer weighed half the weight of the other and the frequency of the note produced by the lighter hammer was doubled, making the note an octave higher.

A little later mathematicians discovered that tones with frequencies in simple ratios could be combined with pleasant effect, but if the ratios were complex then the sound would be discordant. Particular attention was given to the 'tetrachord', which consisted of four tones with frequencies in the ratios 4:5:6:8. By interlinking tetrachords, what we now call a musical scale was formed. Usually called a 'chromatic' scale, a musical scale consists of twelve intervals, called semitones. I shall return to these later.

Computers, when you get right down to the machine's level, only recognise numbers. So it was inevitable that these two worlds would overlap. More than half a century ago, musicians started to experiment with simple tone

generators. These were electric devices that would cause a metal strip to vibrate and formed the basis of the early Hammond organs. Such effects were also employed in science fiction films of the 1950's.

Experimenters also worked with collages of tape recorded sound, which had been dissected into tiny parts and then recombined. Others tried filtering and modulating simple tones. It was not, however, until the coming of the transistor that completely electronic means could be used to generate and shape the tones. Such developments eventually led to the marvellous sound chip incorporated in our microcomputers.

What Do Those Squiggles Mean?

When I was asked to write this series, I gave some thought to what should come first. To program music on the Amstrad computers, you need a knowledge of both music and of the BASIC commands that are used in the program. Of necessity, the manual assumes that you have some knowledge of music and concentrates on the workings of the computer system.

If you have had your computer for a while, and especially,

if you have been reading this magazine for a few months, you should by now have met the sound commands. Perhaps, you have even done some experiments (using my program from the June issue). I therefore decided to concentrate first on deciphering musical notation.

It is not necessary for you to be able to read the notation at a glance. What you do need is to be able to extract from the dots and squiggles, sufficient information to know what the note is, (its pitch), and how long is should sound, (duration). Musical notation also contains other information such as how quickly and how loudly or softly the composer intended the music to be played. We can also use this, but it is not essential at this stage.

Next Time

Next month we shall look at musical notation, including staves, clefs and the various kinds of notes. In the meantime, you can get a head start by typing in Listing 1 and letting the computer show you some of what we are going to cover. This listing will be combined with another next month, so make sure you have it ready before we proceed further.

```

10 GOSUB 1630:GOSUB 1000:GOSUB 1
   760:GOSUB 2480:GOSUB 1890
30 PAPER 0:PEN 1:WINDOW#1,1,40,2
   4,25:CLS#1:WINDOW SWAP 1,0:EN
   D
1000 ,
1010 REM **** Staves ****
1020 ,
1030 BORDER 4:MODE 1
1040 bs=" abcdefgABCDEF":cs="a cd
      fga CD":ds="b de gab DE":p=1
1050 INK 0,0:INK 1,26:INK 2,12:INK
      3,2:PEN 1:PAPER 2:CLS
1060 MOVE 9,378:DRAW 619,0,1
1070 FOR i=1 TO 2
1080 MOVER 0,-18:DRAW -619,0:MOVE
      R 0,-18:DRAW 619,0
1090 NEXT
1100 MOVER 0,-36:DRAW -619,0
1110 FOR i=1 TO 2
1120 MOVER 0,-18:DRAW 619,0:MOVE
      R 0,-18:DRAW -619,0
1130 NEXT
1140 DRAW 0,180:MOVER 619,0:DRAW
      0,-180
1150 ,
1160 , * Print Clef Signs *
1170 ,
1180 PEN 3:PRINT CHR$(22);CHR$(1);
1190 LOCATE 3,1:PRINT CHR$(232);CH
      R$(233);

```

```

1200 LOCATE 3,2:PRINT CHR$(234);CH
      R$(235);
1210 LOCATE 2,3:PRINT CHR$(226);CH
      R$(227);
1220 LOCATE 2,4:PRINT CHR$(238);CH
      R$(239);CHR$(240);
1230 LOCATE 2,5:PRINT CHR$(241);CH
      R$(242);CHR$(243);
1240 LOCATE 2,6:PRINT CHR$(244);CH
      R$(245);CHR$(246);
1250 LOCATE 2,7:PRINT CHR$(247);CH
      R$(248);
1260 LOCATE 2,9:PRINT CHR$(210);CH
      R$(211);CHR$(212);CHR$(213);
1270 LOCATE 2,10:PRINT CHR$(214);C
      HR$(128);CHR$(215);CHR$(216);
1280 LOCATE 3,11:PRINT CHR$(217);C
      HR$(218);
1290 LOCATE 2,12:PRINT CHR$(219);C
      HR$(220);
1300 LOCATE 2,13:PRINT CHR$(221);
1310 PRINT CHR$(22);CHR$(0);
1320 ,
1330 , * Complete Staves *
1340 ,
1350 MOVE 80,378:DRAW 0,-72:MOVER
      4,0:DRAW 0,72
1360 MOVE 80,270:DRAW 0,-72:MOVER
      4,0:DRAW 0,72
1370 RETURN
1600 ,

```

```

1610 REM **** Define Symbols ****
1620
1630 RESTORE:SYMBOL AFTER 210
1640 WHILE chrnum>-1
1650 READ chrnum,sym1,sym2,sym3,sym4,sym5,sym6,sym7,sym8
1660 IF chrnum=-1 THEN 1680
1670 SYMBOL chrnum,sym1,sym2,sym3,sym4,sym5,sym6,sym7,sym8
1680 WEND
1690 KEY DEF 17,1,91,91,129:KEY DEF
1700 F 19,1,93,93,130
1710 KEY 129,CHR$(254):KEY 130,CHR
1720 $(255)
1730
1740 KEY DEF 13,1,49,49,49:KEY DEF
1750 14,1,50,50,50
1760 RETURN
1770 REM **** Stave Description **
1780 **
1790 PEN 0:LOCATE 12,15:PRINT"STAF
1800 F OR STAVE"
1810 LOCATE 2,17:PRINT"A stave, al
1820 so called a staff, consists"
1830 LOCATE 2,18:PRINT"of 5 hori
1840 zontal lines on which music"
1850 LOCATE 2,19:PRINT"is written.
1860 Usually staves are grouped"
1870 LOCATE 2,20:PRINT"in pairs, a
1880 s shown above. The signs at"
1890 LOCATE 2,21:PRINT"the beginn
1900 ng of each stave are called"
1910 LOCATE 2,22:PRINT"clef signs.
1920 The top clef is the treble"
1930 LOCATE 2,23:PRINT"and the low
1940 er one is the bass clef."
1950 t=TIME:WHILE TIME<t+4000:WEND
1960 RETURN
1970 REM **** Types of Notes ****
1980
1990 PEN 0:LOCATE 12,15:PRINT"TYPE
2000 S OF NOTES"
2010 LOCATE 2,17:PRINT"The longest
2020 common note is a semibreve"
2030 LOCATE 2,18:PRINT"and is equi
2040 valent to duration '160'. A"
2050 LOCATE 2,19:PRINT"minim equal
2060 s duration '80'. A crotchet"
2070 LOCATE 2,20:PRINT"equals '40'
2080 , a quaver '20' and a semi-
2090 LOCATE 2,21:PRINT"quaver '10'
2100 . A dot placed after a note"
2110 LOCATE 2,22:PRINT"increases t

```

```

he duration by half.
1960 LOCATE 2,23:PRINT SPC(38)
1970 t=TIME:WHILE TIME<t+4000:WEND
1980 RETURN
1990
2000 REM **** Draw Notes ****
2010
2020 MOVE x,y:DRAW 6,0:DRAW 0,2:
2030 DRAW 2,0:DRAW 0,2:DRAW 2,0
2040 DRAW -2,2:DRAW -2,0:DRAW 0
2050 ,2:DRAW -6,0:DRAW 0,-2
2060 DRAW -2,0:DRAW 0,-2:DRAW -
2070 2,0:MOVER 2,-2:DRAW 2,0
2080 RETURN
2090
2100 MOVE x,y+2:DRAW 6,0:MOVER 2,
2110 2:DRAW -10,0:MOVER 0,2:DRAW 8,0
2120 RETURN
2130
2140 MOVE x+8,y+4
2150 DRAW 0,28:MOVER 1,0:DRAW 0,
2160 -28:MOVER 1,0:DRAW 0,28
2170 RETURN
2180
2190 MOVE x+10,y+30
2200 DRAW 2,-2:DRAW 2,-2:DRAW 2
2210 ,-2:DRAW 2,-2
2220 MOVER -8,10:DRAW 2,-2:DRAW
2230 2,-2:DRAW 2,-2:DRAW 2,-2
2240 RETURN
2250
2260 MOVE x-3,y-26:GOSUB 2110
2270 RETURN
2280
2290 MOVE x+1,y-26
2300 DRAW 2,2:DRAW 2,2:DRAW 2,2
2310 :DRAW 2,2
2320 MOVER -8,-6:DRAW 2,2:DRAW 2
2330 ,2:DRAW 2,2:DRAW 2,2
2340 RETURN
2350
2360 MOVE x+1,y-18:GOSUB 2260
2370 RETURN
2380
2390 REM **** Notes ****
2400
2410 ORIGIN 0,0,0,639,0,399:PLOT 6
2420 40,400,3
2430 y=292:x=130:GOSUB 2020
2440 y=310:x=224:GOSUB 2020:GOSUB
2450 2100

```

```

2510 Y=328:X=324:GOSUB 2020:GOSUB
2070:GOSUB 2100
2520 Y=346:X=426:GOSUB 2020:GOSUB
2070:GOSUB 2220:GOSUB 2250
2530 Y=364:X=534:GOSUB 2020:GOSUB
2070:GOSUB 2220:GOSUB 2250:GO
SUB 2300
2540 PAPER 2:PEN 3:PRINT CHR$(22);
CHR$(1);
2550 LOCATE 9,10:PRINT "(d)":LOCAT
E 7,12:PRINT"semibreve"
2560 LOCATE 15,9:PRINT "(f)":LOCAT
E 14,11:PRINT"minim"
2570 LOCATE 21,8:PRINT "(A)":LOCAT
E 19,10:PRINT"crotchet"
2580 LOCATE 27,7:PRINT "(C)":LOCAT
E 25,9:PRINT"quaver"
2590 LOCATE 34,6:PRINT "(E)":LOCAT
E 29,8:PRINT"semiquaver"
2600 PRINT CHR$(22);CHR$(0);:RETUR
N
2610 ,
2620 REM **** Symbol Data ****
2630 ,
2640 DATA 210,0,0,0,1,2,4,8,16,211
,0,0,255,3,0,0,0,0,212,0,0,12
8,224,224,112
2650 DATA 56,56,213,0,0,0,0,0,96,2
40,240,214,96,240,240,96,0,0,
0,0,215,28,28
2660 DATA 28,28,28,28,28,24,216,96
,0,0,0,96,240,240,96,217,0,0,

```

```

0,0,0,0,1,3
2670 DATA 218,24,48,48,96,192,192,
128,128,219,0,0,0,0,0,0,1,2
20,3,6,12,24
2680 DATA 48,96,192,0,221,2,4,8,16
,0,0,0,0,226,0,0,0,0,1,3,6,12
,227,24,48,112
2690 DATA 208,144,16,16,16,230,7,3
1,48,48,96,96,240,241,231,0,1
92,96,96,51,51
2700 DATA 24,24,232,0,0,0,0,7,8,8,
16,233,0,0,0,0,0,128,64,64,23
4,16,16,16,17
2710 DATA 17,18,20,28,235,64,128,1
28,0,0,0,0,238,24,24,48,48,
96,96,96,192
2720 DATA 239,16,16,16,16,63,80,14
4,16,240,0,0,0,0,128,96,16,24
,241,192,192
2730 DATA 192,192,192,192,96,96,24
2,16,16,16,16,16,16,16,16,243
,8,12,6,6,6,6
2740 DATA 6,6,244,48,48,16,24,12,7
,1,0,245,16,16,16,16,16,16,25
5,16,246,14,12
2750 DATA 28,24,48,64,128,0,247,0,
0,1,3,3,3,1,0,248,16,16,208,2
08,240,224,192
2760 DATA 0,254,0,8,44,56,108,56,1
04,32,255,0,0,32,32,44,52,40,
48,-1,0,0,0,0
2770 DATA 0,0,0,0,0

```

Competition Result - Class Three - Educational Software

You would have gathered by now that the number of Competition entries caught us by surprise. We had planned to announce all the winners in this issue but ran out of time. Why is it taking so long? For a start, each entry has to be given a fair go with a full test, and sometimes requires a professional overseer to assist in the assessment - this is especially relevant in the business section. The Educational class was a little disappointing. Some entries lacked originality, some provided an hypnotic effect with their flashing borders, some had syntax errors, some had hardly any documentation and some made the short list. Our next task is to sift through the short list of the other classes and identify the winners.

CONGRATULATIONS to Mark Snoxell of Upper Ferntree Gully for his entry in Class 3 "SCIENCE TUTOR" Mark wins a DDI-1 Disc Drive

A Case for 'Arnold'

by Robin Nicholas

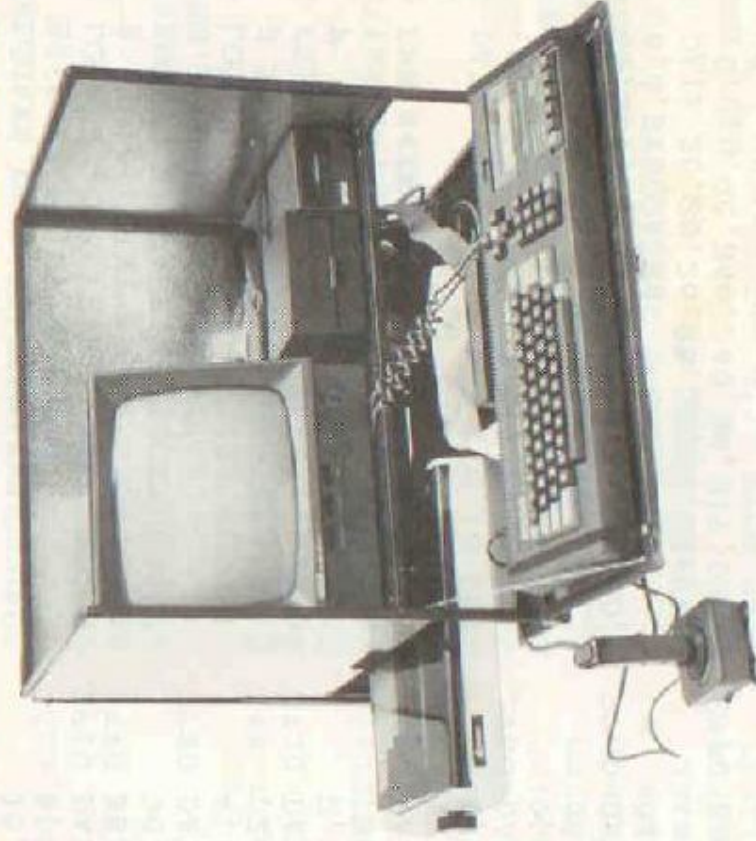
In our house we call the time before we obtained the case for Arnold, B.C. (Before Case). As a member of an Amstrad User Group, and a visitor to other Groups in our City, the effort required to pack up Arnold, unplug the printer, joystick, disc drive, etc, carry the monitor under one arm, keyboard under the other, out to the car with that lot, back for the Disc Drives, discs, joystick, etc, and the same when we arrive at the meeting, sometimes made us wonder if it was worth going at all.

But now, A.C. (After Case) - we just unplug the powerboard from the wall, fold the cord into the case, fold up the keyboard flap and the covers, pick up the case and carry it to the car. On arriving at the meeting Arnold is up and running in less than a minute. (And with the latest model of the case it will even accommodate the average sized printer).

Sounds too good to be true doesn't it? Well let me tell you it is as good as it sounds!

The case has a frame made of light gauge steel tubing which is painted flat black and covered with an embossed aluminium sheeting. The monitor and ancillary shelf is constructed of 3mm masonite, as is the base of the case. The case itself weighs less than 10kg. and can accommodate Monitor, Keyboard, Two Disc Drives, and Joy Sticks, with a standard printer fitting under the ancillary shelf.

It is finished with rubber grip strips on the sides as it is carried by gripping each side of the case forward of the centre and under the ancillary shelf. (The earlier models were fitted with handles but these were found to be inconvenient when carrying the case through doorways).



The case is 24" wide, 18" deep and 20" high. The Monitor sits on a shelf 5" above the keyboard level, a rather handy height for the eye. The keyboard shelf folds down and is supported by a restrainer allowing the case to be placed on a desk that would otherwise be too narrow, and allows the keyboard angle to be increased thus improving the key rake. Needless to say the restrainer is adjustable.

There is ample storage space under the ancillary shelf for joysticks etc. and if there is only one Disc Drive (or two 3" on top of each other) there is ample storage space beside the monitor for Manuals to stand up for easy access.

The Case will accommodate any of the AMSTRAD CPC range but the drawback is that with the weight of the Colour monitor, two drives and printer,

it was just too much for one of the small framed members of our User Group to handle. The manufacturer tells me there is little that can be done about this problem as even halving the weight of the case (at the expense of strength and cost) would not alleviate the problem.

Another slightly annoying problem is the painted frame has a tendency to scratch easily at the folding joints, but as Jimmy Durante used to say "It ain't a perfect world".

The Case retails at \$119.00 (User Group Members can obtain a 10% discount).

Further details may be obtained from the manufacturer:

DONL ENGINEERING, 535 Albion Street, West Brunswick, Victoria 3055, Phone: (03) 383 1498.

JUNIOR JOTTERS

A Column for Young
Amstrad Users

BARCHARTS

from Brendan Piner

This is a program which draws up a Histogram from data which is entered. The computer asks you the year in which the data came from then it asks for the data of the 12 months for that year. After all the data is entered, it draws up a 3D Histogram

```

10 REM *****
*****
20 REM ** Barcharts ** By ** Bre
ndan Piner **
30 REM *** Amstrad CPC464 **** (
1985) *****
40 REM *****
*****
50 MODE 1:BORDER 3:INK 0,3:PAPER
0:INK 1,11:INK 2,3:INK 3,3:C
LS
60 PEN 3:LOCATE 1,25:PRINT" Bar
charts";
70 FOR x%=0 TO 312
80 FOR y%=0 TO 16 STEP 2
90 IF TEST(x%,y%) THEN PLOT 11+x
%*3,348+y%*3,3:PLOT 11+x%*3,3
50+y%*3,1:PLOT 11+x%*3,352+y%
*3,1:PLOT 11+x%*3,354+y%*3,1
100 NEXT:NEXT
110 LOCATE 1,25:PRINT"Amstrad CP
C464 ";
120 LOCATE 17,6:PEN 1:PRINT" For
the"
130 FOR x%=0 TO 304
140 FOR y%=0 TO 16 STEP 2
150 IF TEST(x%,y%) THEN PLOT 205+
x%,242+y%*2,2:PLOT 205+x%,244
+y%*2,2:PLOT 205+x%,246+y%*2,
1
160 NEXT:NEXT
170 a=0:PEN 3
180 LOCATE 1,25:PRINT" Written B
y ":PEN 2
190 FOR x%=0 TO 220 STEP 2
200 FOR y%=0 TO 16 STEP 2
210 a=a+2

```

```

220 IF TEST(x%,y%) THEN PLOT (x%+
a)+105,172+y%*2,3:PLOT (x%+a)
+105,174+y%*2,3
230 NEXT:a=a-16:NEXT
240 LOCATE 1,25:PEN 3:PRINT " Bre
ndan Piner":PEN 3
250 FOR x%=0 TO 312
260 FOR y%=0 TO 16 STEP 2
270 IF TEST (x%,y%) THEN PLOT x%*
2.5+15,52+y%*3,3:PLOT x%*2.5+
15,54+y%*3,3:PLOT x%*2.5+15,5
6+y%*3,3:PLOT x%*2.5+15,58+y%
*3,1
280 NEXT:NEXT
290 LOCATE 1,25:PRINT SPC(30)
300 INK 3,16:PEN 1:INK 2,1:LOCATE
8,24:PRINT "Press <SPACE> to
continue"
310 js=INKEY$:IF (js<>" ") THEN 3
10
320 SOUND 1,500,20,7:GOTO 800
330 REM ** main part of program *
*
340 MODE 1:BORDER 3:INK 0,3:PAPER
0:CLS
350 INK 3,26:PEN 3:INPUT"What yea
r":yy:SOUND 1,500,20,7
360 IF yy<1 THEN CLS:PRINT"Pardon
!":FOR ff=1 TO 2000:NEXT ff:G
OTO 350
370 ORIGIN 55,110
380 DIM d(20)
390 FOR j=1 TO 12
400 PEN 3:LOCATE 1,5:PRINT j:LOCA
TE 4,5:INPUT" month":v:CLS
410 IF v<0 THEN PRINT"No negative
s, Please enter again":SOUND
1,600,20,7:GOTO 400
420 n=j:d(j)=v
430 IF v>n THEN m=v
440 NEXT j
450 FOR q=1 TO n
460 y=d(q)/m*250
470 BORDER 13:INK 0,13:PAPER 0
480 INK 3,0:PEN 3:LOCATE 33,6:PRI
NT "MONTHLY"
490 LOCATE 35,8:PRINT "BAR":LOCAT
E 34,10:PRINT"CHART"
500 LOCATE 33,12:PRINT "of";yy
510 PEN 3:LOCATE 7,20:PRINT"J F M
A M J J A S O N D"
520 LOCATE 7,21:PRINT"A E A P A U
U U E C O E"
530 LOCATE 7,22:PRINT"N B R R Y N
L G P T V C"

```

```

540 GOSUB 590
550 NEXT q
560 PEN 2
570 LOCATE 7,25:PRINT "Press <SPACE> to continue":js=INKEY$:IF (js<>" ")THEN 570
580 SOUND 1,500,20,7:GOTO 690
590 FOR p=0 TO y
600 PLOT 32*q,p
610 INK 2,21:DRAW 30,0,2
620 INK 3,0:DRAW 10,10,3
630 NEXT p
640 FOR p=1 TO 10
650 PLOT 32*q+p,y+p
660 INK 1,9:DRAW 30,0,1
670 NEXT p
680 RETURN
690 BORDER 0:INK 0,0:PAPER 0:INK 1,26:PEN 1:CLS:PRINT"Would you like to enter some more data into the program.Y/N"
700 k$=UPPER$(INKEY$)
710 IF k$="" OR (k$<>"Y" AND k$<>"N")THEN RUN 700
720 IF k$="Y" THEN GOTO 340 ELSE PRINT:PRINT:PRINT"Bye bye!!":END
800 MODE 1:INK 0,0:BORDER 1:PAPER 0:INK 1,26:INK 2,9:INK 3,6:CLS
810 PEN 1:LOCATE 16,1:PRINT"Barch arts"
820 PEN 3:LOCATE 10,4:PRINT"By Brendan Piner (1985)"
830 PRINT:PRINT:PRINT
840 PEN 2:PRINT" This is a program which draws-up a Histogram. The computer asks you the year in which your data comes from."
850 PRINT:PRINT:PRINT:PRINT" Then it asks for the data from the twelve months of that year. After all that is done it draws up a histogram."
860 PEN 1:LOCATE 10,24:PRINT"Press <space> to start"
870 js=INKEY$:IF (js<>" ") THEN 870
880 SOUND 1,500,20,7
890 GOTO 330

```

GRAPHICS DEMONSTRATION

from P. Mezzavia

The following program is a graphics demonstration that produces a similar effect to "string and nail" abstract artforms. I am 13 years old and I think this will create interesting effects on the new GP-700 printer. To obtain a new display press any key. I use my Amstrad for many things but mostly for lists and reminders with Masterfile and Amword.

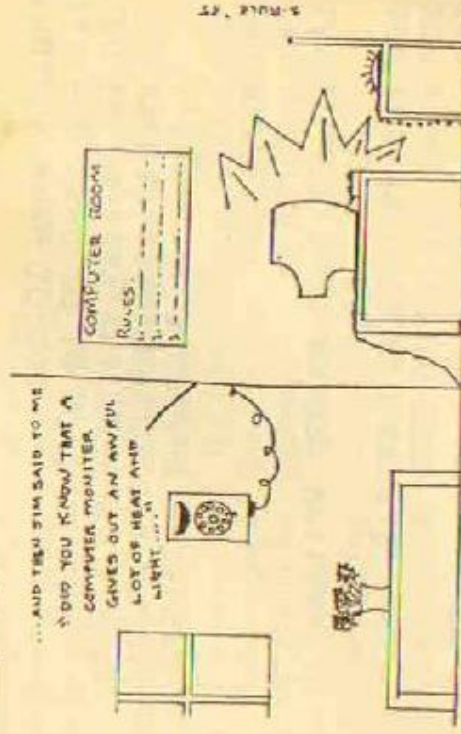
```

10 REM GRAPHLOT.1
20 REM by P. Mezzavia
30 REM <c> 1985
40 MODE 2
50 INK 2,0
60 INK 3,6
70 BORDER 1
80 CLS
90 b%=RND*5+1
100 c%=RND*5+1
110 ORIGIN 320,200
120 FOR a=0 TO 1000 STEP PI/30
130 x%=100*COS(a)
140 MOVE X%,X%
150 DRAW 200*COS(a/b%),200*SIN(a/c%),3
160 IF INKEY$<>" " THEN 80
170 NEXT
180 GOTO 80

```

CARTOON

Stephen Rule sent us this cartoon and says "In case you don't get it (like my Mum didn't) in the second frame, he is supposed to have melted". Good luck with your art class competition.



Bughunt

A Game from Andre Urankar

BUGHUNT owes its existence to an old game called Mugwump. In the original version only text responses were provided. However this newer version provides a graphics field to indicate the 'hits' and 'misses'.

The hunt is based on the application of the mathematical formula for the hypotenuse of a triangle. Playing instructions are simple and are included in the body of the program.

The program has been written to allow modifications to:

The number of guesses allowed - at line 9500
 The number of Bugs - at line 9500
 "Firing" sound - subroutine at 140
 "Hit" sound - subroutine at 150

The Bug was designed by daughter #1, and is held by strings OK\$(1) and OK\$(2) being the top and bottom halves. Good hunting.

1 'Program Name: BUGHUNT
 2 'Developed By: A.M. Urankar
 3 'Date: May 1985
 4 ,
 5 ,

9 GOTO 9000

100 FOR p=1 TO total.bugs: IF x=horiz(p) AND y=vert(p) THEN horiz(p)=0: PRINT#4, "CAUGHT #"

104 NEXT: RETURN
 109 ,

140 ENV 1,15,-1,4: SOUND 1,0,60,15,1,0,1: RETURN

150 FOR ton=50 TO 300: SOUND 1,ton,1: NEXT: RETURN

999 ,

1000 RANDOMIZE TIME

1002 CLS#1: CLS#2: CLS#3: CLS#4: FOR x=1 TO 10: PRINT#1, line.1\$: PRINT#1, line.2\$: NEXT: PRINT#3, "1

2 3 4 5 6 7 8
 9 10": PRINT#2, "1 2 3 4 5 6 7 8 9 10": PRINT#6, STRING\$(25, " "): PRINT#5, "'BUG' Hunt"

1008 FOR v=1 TO 10: FOR h=1 TO 10: position(v,h)=0: NEXT: NEXT
 1010 FOR x=1 TO total.bugs
 1011 horiz(x)=INT(RND(1)*10): IF horiz(x)=0 THEN 1011
 1012 vert(x)=INT(RND(1)*10): IF vert(x)=0 THEN 1012
 1013 position(horiz(x), vert(x))=1
 1020 NEXT

1024 guess=1: bugs=total.bugs
 1025 WHILE bugs>0 AND guess<max.guess+1
 1027 ok=0

1030 PRINT#4, "Guess "USING"##"; guess;: INPUT#4, " "; y\$, x\$: x=VAL(x\$): y=VAL(y\$): IF x=0 OR x>10 OR y=0 OR y>10 THEN PRINT#4, "Try Again!!" CHR\$(7): GOTO 1030
 1035 IF position(x,y)=-1 THEN PRINT#4, "Try Again!!": GOTO 1030
 1038 guess=guess+1

1040 LOCATE#1, x*2-1, y*2-1: GOSUB 140
 1045 IF position(x,y)=1 THEN ok=1:

GOSUB 100: bugs=bugs-1
 1050 position(x,y)=-1
 1055 IF ok=1 THEN PRINT#1, ok\$(1) ELSE PRINT#1, ok\$(0)

1060 LOCATE#1, x*2-1, y*2:
 1065 IF ok=1 THEN PRINT#1, ok\$(2): GOSUB 150 ELSE PRINT#1, ok\$(3)

1070 FOR q=1 TO total.bugs: IF horiz(q)=0 THEN 1080
 1075 dis=SQR((horiz(q)-x)^2+(vert(q)-y)^2): PRINT#4, "q"at"USING"##.##"; dis;: PRINT#4, " units";

1080 NEXT: PRINT#4
 1100 WEND: IF bugs=0 THEN 1300
 1105 CLS#4: PRINT#4, "Sorry!! You have just RUN OUT of guesses!": PRINT#4, "": PRINT#4, " This is where those tricky 'BUGS' were hiding...": PR

Calendar Generator

from Arthur Harris

Calendar is a delightful piece of trivia which some people will find very useful. It displays on the screen a calendar for any month. It also allows you to move from month to month forwards and backwards and from year to year, both forwards and backwards.

The crucial part of the program is the menu, which is printed below.

```

For NEXT MONTH      Press ' '
For PREV MONTH      Press ' '
For RE-START        Press 'R'
For JANUARY          Press '1'
For FEBRUARY         Press '2', etc.
For OCTOBER          Press 'O'
For NOVEMBER         Press 'N', etc.
For NEXT YR.         Press ' '
For PREV YR          Press ' '
    
```

Given the number of different printers being used on computers generally, and the Amstrad in particular, I have not attempted to produce a version that will print out the calendar. I will leave that as an exercise for the readers. I have produced a version of a screen dump program suitable for a C-Itoh 8510A printer which dumps a screen image in 121 seconds, by using a combination of machine language and BASIC (the same program using only BASIC took nearly an hour). One of the early Amstrad CPC-464 magazines, from England, included a machine language screen dump program for the DMP-1 and for an Epson printer. I would be very interested to know whether it works.

The alternate method to explore is to dissect how the program places the image on the screen and format the same image out to the printer.

I hope this program is of use to readers, as apart from giving a calendar, if they dissect the program, they will find the routines to find which day of the week any day fell on (or will fall on, in the future).

```

10 ' Calendar Generation Routine
20 ' m$( ) --Months
30 ' m( ) --Number of Days in Mon
   th
40 ' q$( ) --Days of Week
50 ' *****
60 e1$=STRING$(37, "*"): MODE 1
    
```

```

70 DIM m$(12), m(12), q$(6)
80 q$=" 1 2 3 4 5 6 7 8 9 10 11 12 1
31 41 51 61 71 81 91 101 111 121
28 29 30 31"
90 CLS:PRINT TAB(2);e1$:PRINT TA
B(11);"PERPETUAL CALENDAR":PR
INT TAB(2);e1$:PRINT:PRINT TA
B(5);"This program will print
out a":PRINT TAB(5);"complet
e calendar for any year."
100 PRINT:PRINT TAB(7);"For NEXT
MONTH - Press ' '":PRINT TAB(
7);"For PREV MONTH - Press '
'":PRINT TAB(7);"For RE-START
- Press 'R'"
110 PRINT:PRINT "For JANUARY-Pre
s '1', FEBRUARY '2', etc."
120 PRINT "For OCTOBER-Press 'O',
NOVEMBER 'N', etc."
130 PRINT:PRINT "For NEXT YR-Pre
s ' ', For PREV YR '-'"
150 ' READ IN MONTHS
160 RESTORE 170:FOR i=1 TO 12:REA
D m(1), m$(1):NEXT i
170 DATA 31, JANUARY
180 DATA 28, FEBRUARY
190 DATA 31, MARCH
200 DATA 30, APRIL
210 DATA 31, MAY
220 DATA 30, JUNE
230 DATA 31, JULY
240 DATA 31, AUGUST
250 DATA 30, SEPTEMBER
260 DATA 31, OCTOBER
270 DATA 30, NOVEMBER
280 DATA 31, DECEMBER
290 PRINT:INPUT "Year " ; y;
INPUT "Month (1-12)"; m1: IF m1
*(13-m1)<1 THEN 290
300 ' START OF GENERATION ROUTINE
310 FOR i=1 TO 6:q$(i)=STRING$(10
0, " "):NEXT i
340 m=m1:d=m(m)+(1-SGN(y/4-INT(y/
    
```

(continued on Page 32)

Book Review

by Simon Anthony

I'll let you into a secret. 'A child's guide to the Amstrad Micro' is not necessarily a *child's* guide to the Amstrad micro. I have it on good authority that a number of adults have been buying copies, insisting that they be wrapped, and smuggling them into their home. No - not as birthday presents for children, but as a tutorial for themselves!

What a shame they have to go to such lengths to get hold of a book which introduces the Amstrad to the beginner, but alas there are few, if any, that start at such a basic level.

Invariably, books for beginners assume certain things. Not this one - it really starts at the beginning.

'A Child's guide to the Amstrad Micro' comes from the Cambridge University Press stable of other 'child's guides'. But unlike publications which have tried to cash in on the Amstrad success by rushing out, for example, an old Commodore reprint with a few changes to make it look as though it has been written specifically for the Amstrad, this book appears to have been written from scratch.

Of course, the book caters primarily for children, and as such is cleverly presented in five sections by 'experts', all of whom have names derived from the word COMPUTER.

Pru Comet is the typist and shows the ways in which the various types of keys work and the position of the keys on the keyboard. Throughout her tour she provides simple typing tasks with pictures of the results expected on the screen.

P.C. Truemo is the investigator. He shows how to get started on the computer, how to make it print



culminating in five projects which put to use the previous examples. The projects are entertaining yet clearly devised to make the reader think about the structure and content of the programs being written. They feature 'knock, knock' jokes, and 'see who ends up with the last counter' game, music and a simple strategy game. Naturally enough, this is the largest section of the five.

The fifth and last section is a reference section, handled by Ms. O.C. Termup. In effect it contains a glossary of the terms used in the book with references to the CPC User Guide that comes with each machine.

It is clear that the overall structure and presentation of this introduction to the world of Amstrad computing is novel and should be appealing to youngsters from about 8 years and upwards. For older people who just cannot get to grips with the "mysteries" of computing it is recommended, even if it's brought home in a brown-paper bag!

numbers and words, calculate answers and finally store items in its memory. "INVESTIGATE" - track down every possible mistake until you get everything correct" is his advice, and provides more tasks with sample screen results.

Mort Puce is the artist, and he is used to show how to draw coloured pictures on the screen and produce sound. This section is a gentle introduction to the DRAW, PLOT and MOVE commands along with BORDER, PEN, PAPER and INK. Sound gets allocated just one page - too little for such a large subject. The fourth section is conducted by Professor O. Crumpet - the designer who shows how to copy, adapt and finally design programs. We are now half way through the book and just about to enter our first program. This is not a criticism but an indication of what I said earlier - the book really does cover the basics so often forgotten.

Sample programs are printed to provide example of the use of various commands, operators and signs,



Some hints from Murray Bridge ...

No, Murray Bridge is not the author but a place in South Australia from where Lindsay Allen hails and who takes the credit for supplying these useful hints.

HINT 1

When manipulating strings with MID\$, LEN, LEFT\$ etc., it is possible to get an 'improper argument' error message. If the function is a legal one, then what Arnold is trying to tell you is that the string does not exist.

The easiest way to avoid this problem is to test that the string does exist before manipulating it, for example:

```
50 IF test$=" " THEN 60 ELSE PRINT LEN(test$)
```

HINT 2

If you are using a WHILE - WEND loop, the test is only done at the beginning of the loop. If you want the loop to stop the moment the test value changes, you will have to do something like this:

```
60 j=1:WHILE name$<>"END"
70 INPUT name$(j):IF name$(j)="END" THEN j=j+1
; GOTO 90
80 INPUT age(j):j=j+1
90 WEND
```

If the test was not done in line 70, END and an "age" would have been added to your list of names.

HINT 3

You may have discovered a small problem in the arithmetic routine when using decimals. For example, if you enter:

```
120.23 - 120 the answer displayed is 0.229999998 !!
```

Now while this is a very small error, over a large program it could add up to a significant amount. The cure is simple, use the ROUND function, for example:

```
answer=120.23 - 120:answer=ROUND(answer,7): ?
```

which will provide an answer of .23.

HINT 4

While experimenting with my Amstrad CPC464, I discovered that if you save a file to tape using OPENOUT so that it was written the same format as a Basic listing, the computer would reload it as a Basic program using LOAD or CHAIN MERGE.

This could be used in a number of ways. For example, the program below could be added to a character defining program.

```
10 REM **Saving File in Basic Format**
20 DIM b(255,8):INPUT"SYMBOL AFTER";sym:
symstart=sym
30 PRINT"Input data for SYMBOL "sym
40 FOR z=1 TO 8
50 INPUT;b(sym,z):IF b(sym,z)>255 OR
```

```
b(sym,z)<0 THEN PRINT CHR$(7):GOTO 50
60 NEXT z
70 PRINT"Do you want to cont? (y/n)"
80 a$=INKEY$:IF a$="y" then sym=sym+1:GOTO 30
90 IF a$<>"n" THEN 80
100 INPUT"What Basic line no. to start on",bln
110 INPUT"What program name",prname$
120 OPENOUT prname$
130 FOR q=symstart to sym
140 PRINT#9,bln" SYMBOl ";;WRITE#9,q,b(q,1),
b(q,2),b(q,3),b(q,4),b(q,5),b(q,6),b(q,7),b(q,8)
150 bln=bln+10:NEXT q
160 CLOSEOUT
170 PRINT"Finished":END
```

Here is an example of what the program can produce.

```
10000 SYMBOl 140,23,56,89,2,5,68,59,54
10010 SYMBOl 141,12,45,2,3,6,9,89,45
10020 SYMBOl 142,6,3,0,0,2,3,63
10030 SYMBOl 143,85,45,1,0,0,1,1,255
```

HINT 5

The Amstrad numeric keypad has no comma, which can be annoying when entering data, SYMBOLs or SOUND. The following will set the "." key to a "," while the shift key is down. If the Caps Lock is pressed while the Ctrl key is held down, it will be locked to a ".".

```
10 KEY DEF 7,0,46,44,44
```

HINT 6

To reset the screen ready for listings, the "." key can again be used in conjunction with a depression of the Ctr key.

```
10 KEY DEF 7,0,46,44,138:KEY 138,CHR$(13)+
"MODE 2:INK 0,1:INK 1,24:PEN 1"+CHR$(13)
```

Another handy change is:

```
10 KEY DEF 18,0,139,139,141:KEY 141,CHR$(13)
+"RUN"+CHR$(13)
```

This sets the large enter key to RUN the program when it is pressed while holding the Ctrl key down. (Note: if you use this, do not alter Key 139).

... and one from Fraser

For those of you who possess both a colour monitor and a disc drive, here is a hint from Chris Rogers of Fraser in the ACT to save you going blind every time you invoke CP/M.

The colours selected as standard for operating under the control of CP/M are defined in a CP/M table to which the user has easy access. The CP/M command utility SETUP.COM allows the user to vary all sorts of details relating to the CP/M operating environment. At present, I

(continued on Page 32)

Super-Quiz Program

from Carl Allen

Super Quiz currently has 20 set questions, but is structured so that it can be easily expanded. Once past the title screen and instruction page, you are asked to input your name. Simple enough? OK, then comes the hard part. The questions are read one after the other. If you type in the correct answer you will be rewarded with an encouraging, high pitched beep. On entering the wrong answer, a deep rumble will be heard.

The questions run in a set format, so if you are going to add more questions, this is what you enter:

```
PEN 1:PRINT"QUESTION (Question number here)"
PEN 2:LET q$=" (Put question here) "v=v+1
LET a$=" (Put answer here in CAPITALS)"
GOSUB 2000
```

Of course, this can all be written on one line so as to save room to add more questions and the program re-numbered. This version has approximately 30 lines left for additions. It fully uses Arnold's colour capabilities in mode 1 and is easily played, although the answers have to be entered in capitals. (Press the 'Caps Lock' after the program has loaded). Once completed, the program provides a score and percentage.

```
1 REM *****
  *****
2 REM *
3 REM *
4 REM *
5 REM *
6 REM *
7 REM *
8 REM *
9 REM *****
  *****
```

SUPER QUIK-QUIZ

By Carl Allen

(C) 1985

26/8/85

=====

```
10 CLS:INK 0,0
11 v=0
12 ,
15 REM **** TITLE SCREEN ****
16 ,
20 MODE 0:INK 1,2:INK 2,6:INK 3,
  8:INK 4,16:BORDER 2:PEN 2
21 PLOT 10,10:DRAW 10,390:DRAW 6
  30,390:DRAW 630,10:DRAW 10,10
22 PLOT 20,20:DRAW 20,380:DRAW 6
  20,380:DRAW 620,20:DRAW 20,20
30 LOCATE 6,2:PRINT"QUICK QUIZ"
40 PEN 3:LOCATE 10,5:PRINT"By":L
  OCATE 6,7:PRINT"Carl Allen":L
  OCATE 6,8:PRINT"
  "
50 PEN 2:LOCATE 6,3:PRINT"-----"
  "
60 PEN 1:LOCATE 8,15:PRINT CHR$(
  164)" 1985"
70 PEN 4:LOCATE 5,24:PRINT"HIT S
  PACEBAR":LOCATE 5,25:PRINT"==
  "
80 a$=INKEY$:IF a$<>" " THEN 80
85 GOSUB 3000
90 INK 0,1:GOSUB 1000
980 ,
990 REM ** QUESTION DATA **
991 ,
1000 CLS:MODE 1:INK 1,6:INK 2,23:I
  NK 3,0:BORDER 24
1010 PEN 1:PRINT"WHAT IS YOUR NAME
  ":PEN 3:INPUT z$
1020 PEN 1:PRINT" QUESTION .1"
1030 PEN 2:LET q$="What model Hold
  en came after the FX":v=v+1
1040 LET a$="FJ"
1050 GOSUB 2000
1055 PEN 1:PRINT" QUESTION .2"
1060 PEN 2:LET q$="What river empt
  ies into the ocean near Carn
  arvon, W.A":v=v+1
1070 LET a$="GASCOYNE"
1080 GOSUB 2000
1085 PEN 1:PRINT" QUESTION .3"
```

```

1090 PEN 2:LET q$="What is the largest island in the West Indies.":v=v+1
1100 LET a$="CUBA"
1110 GOSUB 2000
1115 PEN 1:PRINT" QUESTION .4"
1120 PEN 2:LET q$="How many stars are on the American Flag.":v=v+1
1130 LET a$="50"
1140 GOSUB 2000
1145 PEN 1:PRINT" QUESTION .5"
1150 PEN 2:LET q$="What is the capital of Czechoslovakia.":v=v+1
1160 LET a$="PRAGUE"
1170 GOSUB 2000
1180 PEN 1:PRINT" QUESTION .6"
1190 PEN 2:LET q$="Who was the first non-human to win an Oscar.":v=v+1
1200 LET a$="MICKEY MOUSE"
1210 GOSUB 2000
1220 PEN 1:PRINT" QUESTION .7"
1230 PEN 2:LET q$="What is 'The Royal Game'.":v=v+1
1240 LET a$="CHESS"
1250 GOSUB 2000
1260 PEN 1:PRINT" QUESTION .8"
1270 PEN 2:LET q$="What does U.S.S.R stand for":v=v+1
1280 LET a$="UNION OF SOVIET SOCIALIST REPUBLICS"
1290 GOSUB 2000
1300 PEN 1:PRINT" QUESTION .9"
1310 PEN 2:LET q$="What ocean are the Aleutian Islands in.":v=v+1
1320 LET a$="THE PACIFIC"
1330 GOSUB 2000
1340 PEN 1:PRINT" QUESTION .10"
1350 PEN 2:LET q$="Where in London are the Crown Jewels kept.":v=v+1
1360 LET a$="THE TOWER OF LONDON"
1370 GOSUB 2000
1380 PEN 1:PRINT" QUESTION .11"
1390 PEN 2:LET q$="What's a group of kittens called.":v=v+1
1400 LET a$="A KINDLE"
1410 GOSUB 2000
1420 PEN 1:PRINT" QUESTION .12"
1430 PEN 2:LET q$="How many equal sides are there in a celestial triangle.":v=v+1
1440 LET a$="TWO"
1450 GOSUB 2000
1460 PEN 1:PRINT" QUESTION .13"
1470 PEN 2:LET q$="How many balls are used in a game of billiards.":v=v+1
1480 LET a$="THREE"
1490 GOSUB 2000
1500 PEN 1:PRINT" QUESTION .14"
1510 PEN 2:LET q$="What was the name given to the Austrian Olympic swimming team of which Neil Brooks was part.":v=v+1
1520 LET a$="THE MEAN MACHINE"
1530 GOSUB 2000
1540 PEN 1:PRINT" QUESTION .15"
1550 PEN 2:LET q$="What Australian TV comedy brought Ross Higgins and Judy Farr together in the starring roles.":v=v+1
1560 LET a$="KINGSWOOD COUNTRY"
1570 GOSUB 2000
1580 PEN 1:PRINT" QUESTION .16"
1590 PEN 2:LET q$="Who wrote the book 'The Lord of the Rings' and 'The Hobbit'. (Surname Only)":v=v+1
1600 LET a$="TOLKIEN"
1610 GOSUB 2000
1620 PEN 1:PRINT" QUESTION .17"
1630 PEN 2:LET q$="What was the name of the second movie in the great 'Star Wars' trilogy.":v=v+1
1640 LET a$="THE EMPIRE STRIKES BACK"
1650 GOSUB 2000
1660 PEN 1:PRINT" QUESTION .18"
1670 PEN 2:LET q$="What year followed 1 B.C.":v=v+1
1680 LET a$="1 A.D"
1690 GOSUB 2000
1700 PEN 1:PRINT" QUESTION .19"
1710 PEN 2:LET q$="What year did World War I start.":v=v+1
1720 LET a$="1914"
1730 GOSUB 2000
1740 PEN 1:PRINT" QUESTION .20"
1750 PEN 2:LET q$="What two sports use the term 'Ace'.":v=v+1
1760 LET a$="TENNIS AND GOLF"
1770 GOSUB 2000
1780 LET a$="TENNIS AND GOLF"
1790 GOSUB 2000
1800 REM **MUSIC & ANSWER CORRECTIONS**

```

```

1981 ON**
1989 ENT 2,90,-6,2:ENT 1,100,2,2:E
NV 12,10,-2,10:T=500:G=500:FO
R T=0 TO 100:T=T+10:G=G-5:SOU
ND 1,T,5,15,0,2:SOUND 2,T+1,5
,15,0,2:SOUND 3,T,5,15,0,1:NE
XT:SOUND 1,60,70,15,0,1
1990 PEN 2:PRINT z$;" , you got";c;
"correct out of";v:PRINT"That
works out to";c/v*100;"%"
1991 PEN 1:PRINT"
T SPACEBAR "
1992 a$=INKEY$:IF a$<>" " THEN 199
2
1993 GOSUB 4000
2000 PRINT q$
2010 PEN 1:INPUT b$
2020 IF a$=b$ THEN PEN 3:SOUND 1,3
0,30,6:PRINT"CORRECT!!":c=c+
1:RETURN
2030 IF a$<>b$ THEN PEN 3:SOUND 1,
3822,50,7:PRINT"WRONG!! The c
orrect answer is ";a$:RETURN
2989
2990 REM ** INSTRUCTION SCREEN **
2991
3000 CLS:MODE 1:BORDER 0:INK 0,0:I
NK 1,6:INK 2,8
3010 LOCATE 11,2:PEN 1:PRINT"*":L
OCATE 14,2:PEN 2:PRINT"QUICK
QUIZ":PEN 1:LOCATE 25,2:PRINT
"*":PEN 2:LOCATE 14,3:PRINT"
":PEN 1:LOCATE 7,5:
PRINT"INSTRUCTIONS":LOCATE 7,
6:PRINT"
3020 PEN 2:LOCATE 4,8:PRINT"This i
s a general knowledge quiz
which forms as a solid t
est of your I.Q."
3030 LOCATE 4,11:PRINT"The topics
range from literature to
sport and you have to attempt
all
questions. An enco
uraging beep is
in reply to a correct answer
, and a low rumble is heard
when the
wrong answer i
s produced."
3040 LOCATE 4,17:PRINT"At the end
of all questions, you will be
told your score and then will
be given, very accurately, y
our percentage."
3050 LOCATE 1,21:PRINT"A good scor

```

```

e would be one more than 75%"
3060 LOCATE 4,22:PRINT"NB:":LOCATE
7,22:PRINT"Remember to activ
ate the 'CAPS LOCK':":PEN 1:LO
CATE 20,24:PRINT"ood ":PEN 2:
LOCATE 19,24:PRINT"G":LOCATE
24,24:PRINT"L":PEN 1:LOCATE
25,24:PRINT"uck!":LOCATE 3,
25:PRINT"PLEASE PRESS SPACEBA
R TO CONTINUE.."
3070 a$=INKEY$:IF a$<>" " THEN 307
0
3080 RETURN
3090
3098 REM ** AU'REVOIR **
3099
4000 MODE 0:CLS:BORDER 0:INK 1,6:I
NK 2,2,20:INK 3,8
4010 LOCATE 4,4:PEN 3:PRINT"Thanky
ou for":LOCATE 4,5:PRINT"play
ing":PEN 1:PRINT" SUPER QUIK
-QUIZ"
4020 PEN 2:LOCATE 5,15:PRINT"T H E
.E N D":LOCATE 5,16:PRINT"
":PEN 3:LOCATE 5,24:P
RINT"Hit Spacebar":PEN 1:LOCA
TE 10,13:PRINT CHR$(251)
4030 a$=INKEY$:IF a$<>" " THEN 403
0
4037
4038 REM ** REDEFINES **
4039
4040 CLS:MODE 1:INK 1,24:PEN 1:BOR
DER 1:INK 0,1

```



Sorting Methods - Part Two

from Arthur Harris

As mentioned last month, I have written a program which compares five sorting methods - Bubble, Modified Bubble, Insertion, Shell and Quick Sorts. The following listing is that of the program run on the Amstrad and after that is an explanation of each section of the program.

```

10 MODE 2: RANDOMIZE TIME: z$=""###
###.##": DIM x$(1000), x1$(10
00), x2$(1000), x3$(1000), x4$(1
000), s9(20,2)
17 PRINT #8, SPC(6) "Bubble"; SPC
(3) "M/Bubble"; SPC(3) "Insert
tion"; SPC(4) "Shell"; SPC(3)
"Quicksort"; SPC(3) "Number"
; TIME/300
20 FOR z=10 TO 100 STEP 10: FOR i
=1 TO z
30 FOR j=1 TO 5: x$(i)=x$(i)+ CHR
$(RND*26+64): NEXT j
40 x1$(i)=x$(i): x2$(i)=x$(i): x3$(
i)=x$(i): x4$(i)=x$(i): NEXT i
47 b=TIME: x=z
120 s=0: l=1: r=2
150 IF r>x THEN 250
160 IF x1$(r)>x1$(l) THEN 220
170 t$=x1$(r): x1$(r)=x1$(l): x1$(l
)=t$: s=-1: t=1
220 l=l+1: r=r+1: GOTO 150
250 x=t: IF s THEN 120
270 f=TIME: a=FREE (""): g=TIME: i1=1
: j1=z
300 i=i1: j=j1: s=-1
330 IF x$(i)<=x$(j) THEN 380
340 t$=x$(i): x$(i)=x$(j): x$(j)=t$
: s=SGN(-s)
380 IF s=1 THEN i=i+1: GOTO 390
385 j=j-1
390 IF i<j THEN 330
400 IF i+1=j1 THEN 440
410 p=p+1: s9(p,1)=i+1: s9(p,2)=j1
440 j1=i-1: IF i1<j1 THEN 300
460 IF p=0 THEN 510
470 i1=s9(p,1): j1=s9(p,2): p=p-1: G
OTO 300
510 q=TIME: a=FREE (""): h=TIME: x=z

```

```

520 s=0: l=1: r=2
550 IF r>x THEN 650
560 IF x2$(r)>=x2$(l) THEN 620
570 t$=x2$(r): x2$(r)=x2$(l): x2$(l
)=t$: s=-1
620 l=l+1: r=r+1: GOTO 550
650 IF s=-1 THEN 520
670 e=TIME: a=FREE (""): r=TIME: FOR
i=1 TO z-1: t$=x3$(i): c=1: FOR
j=1+1 TO z: IF x3$(j)>t$ THEN
690
680 t$=x3$(j): c=j
690 NEXT j: IF c=i THEN 710
700 x3$(c)=x3$(i): x3$(i)=t$
710 NEXT i: s=TIME: a=FREE (""): d=TI
ME
720 j=z
730 j=INT (j/2): IF j=0 THEN 1000
740 k=z-j: FOR i=1 TO k: l=1
745 m=l+j: IF x4$(l)<=x4$(m) THEN
760
750 t$=x4$(l): x4$(l)=x4$(m): x4$(m
)=t$: l=l-j: IF l>0 THEN 745
760 NEXT i: GOTO 730
1000 t=TIME: u=(f-b)/300: v=(q-g)/30
0: w=(e-h)/300: y=(t-d)/300: x=(
s-r)/300
1010 FOR k=1 TO z: x$(k)="": x1$(k)=
"": x2$(k)="": x3$(k)="": x4$(k)
="": NEXT k
1020 FOR j=1 TO 20: s9(j,1)=0: s9(j,
2)=0: NEXT j
1030 PRINT #8, USING z$; w, u, x, y, v, z
: NEXT z
INITIALISATION
Line 10 sets the USING format and dimensions
the necessary arrays;
Line 17 prints the headings for the table;
Lines 20-30 set the range of list sizes (Z-loop) and
generates the necessary number of
strings of 5 randomly chosen letters from
A to Z inclusive;
Line 40 takes 4 copies of this list, so that each
method is dealing with an identical list.

```

MODIFIED BUBBLE SORT

Line 47 reads the internal clock and sets the initial pass to encompass the complete list;
Line 120 resets the "flag" to false and sets the positions for the first comparison;
Line 150 tests whether a pass through the list is complete;
Line 160 compares the two items and tests whether a swap is needed;
Line 170 makes the swap, sets the "flag" to true and locates the lower of the two items swapped;
Line 220 increments the positions of the items to be compared and returns to pass through the basic algorithm;
Line 250 sets the limit of the un-sorted portion of the list and tests whether any swaps were made on the previous pass.

On my count, this sort occupies 148 bytes in the Amstrad and requires 6 additional variables.

QUICK SORT

Line 270 records the times (end of Modified Bubble Sort and start of Quick Sort) and sets the maximum limits of the list and invokes the "garbage collector";
Line 300 sets the positions of the first items for comparison and sets the "flag" for whether the sort will proceed from the bottom of the list upwards or from the top of the list downwards (it starts by proceeding downwards from the top);
Line 330 compares the two items and tests whether a swap is needed;
Line 340 makes the swap and changes the direction of the sort;
Line 380 tests the "flag" for which direction the sort is to proceed and changes one of the pointers;
Line 385 changes the other pointer (only one of the pointers is changed);
Line 390 tests whether the first pass of the list is complete;
Line 400 tests whether it is time to obtain the limits of the next partition to be dealt with;
Line 410 records the limits of the currently determined partitions and increments the stack pointer;
Line 440 sets the upper limit for the next partition and returns to deal with that partition;
Line 460 test whether the sort is complete;

Line 470

pops off the stack the limits of the next partition to be handled, decrements the stack pointer and returns to deal with that partition.

On my account, this sort occupies 262 bytes in the Amstrad and requires 7 additional variables, plus the stack.

As mentioned earlier, none of the discussions on this sort that I have seen, addresses the problem of the size of stack required for a given size of list. Accordingly, I wrote a program which sorts lists of various sizes and records the number of rows used in the stack. This program is similar to the one described here, except that the strings sorted are only two characters long. Various numbers of replications, from 20 to 90, gave me data to which I could apply statistical analysis. My approach was to find the mean and standard deviation of the sample. Assuming the sample approximates a normal distribution, 99.73% of all values lie within 3 standard deviations either side of the mean. I produced an envelope value by adding 3 times the standard deviation to the mean. I then used the method of least squares to fit a curve to these envelope values.

Using the PC-1500 and my TRS-80 Model I, I covered the range from 5 to 2000 item lists. It proved impossible to cover the full range with a single curve. Good approximation was obtained by a cubic curve for the ranges 5 to 225 and 250 to 2000. This is convenient as one equation can be used for the PC-1500 and both equations can be used in larger machines. In the terminology used in the program described above, Z is the number of items to be sorted, $N = Z/100$ and M is the size of array required for the stack.

(a) for $5 \leq Z \leq 255$,

$$M = \text{INT}(2.22131854 + 13.23168451*N -$$

$$7.310600829*N*N + 1.444030691*N*N*N) + 1$$

(b) for $250 \leq Z \leq 2000$,

$$M = \text{INT}(8.10162567 + 1.585403563*N -$$

$$0.1220688825*N*N + 0.003366138505*N*N*N) + 1$$

These figures are exactly as produced by the PC-1500 and cover the envelope values, except when $Z=185$, where the value of M is 1 too low. Some experimentation is necessary to determine the minimum accuracy of the coefficients which will still give the correct results.

The size of the array varies from 3 when $Z=5$ to 13 when $Z=255$ for equation (a) and from 12 when $Z=250$ to 18 when $Z=2000$ for equation (b).

BUBBLE SORT

Line 510

records the times (end of Quick Sort, start of Bubble Sort) and sets the size of the list and invokes the "garbage collector";
Line 520 sets the "flag" to false and sets the

positions of the items for the first comparison;
 tests whether a pass through the list is complete;
 compares the items and tests whether a swap is needed;
 performs the swap and sets the "flag" to true;
 increments the pointers to the items to be compared and returns to pass through the basic algorithm;
 tests whether a swap was made during the last pass through the list and returns for another pass if this was so.

On my count this occupies 149 bytes in the Amstrad and requires 5 additional variables.

INSERTION SORT

Line 670 records the times (end of Bubble Sort, start of Insertion Sort), sets the limits for the outer loop (1st to last but 1 item), records the current item and its position and sets the limits for the comparison loop and finally tests whether the comparison item needs to replace the current item (the "garbage collector" is also called);
 replaces the current item with the comparison item and records the position of the comparison item;
 ends the comparison loop and tests whether a swap is required;
 performs the swap;
 ends the outer loop and records the times (end of Insertion Sort, start of Shell Sort) and calls the "garbage collector".

On my count this routine occupies 121 bytes in the Amstrad and requires 4 additional variables.

SHELL SORT

Line 720 sets the size of list to be sorted;
 Line 730 halves the interval for the next set of comparisons and tests whether the sort is complete;
 sets the upper limit for the loop for comparisons, sets the limits for the loop and sets the pointer for the first item in the comparison;
 sets the pointer for the second item in the comparison and tests whether a swap is required;
 performs the swap, decrements the pointer to the first item in the

comparison by the interval and then tests whether this position is valid for another comparison;
 ends the loop and returns for the next pass.

On my count this routine occupies 147 bytes in the Amstrad and requires 5 additional variables.

PRINTOUT OF SORTED LISTS

Lines 800-840 a subroutine to print out the sorted lists.
 This was used during the development of the program to check that all the sort routines were performing correctly. It was called from Line 1005. Note the method used to quickly include or exclude this from operation.

TIDY UP

Line 1000 records the time (end of Shell Sort) and obtains the elapsed times for sorting and branches to the printout subroutine if needed;
 sets each of the arrays to null strings ready for the next size of sort;
 zeros the elements of the stack, ready for the next sort;
 Prints out the times taken for each of the sorts for the current size of list.

GENERAL COMMENTS

Three sets of comparison were carried out and all revealed similar trends. The Quick Sort is usually the fastest sort. In a few cases (about one-third and all restricted to the shorter end of the scale) the Shell Sort is marginally faster. The Insertion Sort is next fastest, followed by the Modified Bubble Sort and, slowest of all, is the Bubble Sort.

It should be noted that the times are not monotonically rising. There are reversals of the direction within the times for any given sort routine. This demonstrates the statement that the time taken (used as one measure of efficiency) depends on the initial condition of the list. Obviously, although randomly generated, the initial degree of ordering of some of the lists tended to produce a faster (or slower) time than expected. To produce a definitive answer (which is not really possible, given the number of parameters and the variability of each) would require a large number of replications of the test performed here.

Based on the information presented here, in terms of the number of bytes occupied, the number of additional variables and the time taken and the fact that the initial condition of the list is generally not known, my choice for the ideal sort routine would be the Shell Sort. This is somewhat contrary to my previous claims which were based on what I now

believe to be false claims for a supposedly superior version of the Shell Sort on the TRS-80 Model I. This version was so slow that I think a Bubble Sort would have beaten it. I still maintain that the Quick Sort is the fastest generally but it suffers the disadvantage of the larger number of bytes required, the extra additional variables and the artificial stack required.

I shall probably perform the number of replications necessary and perform the statistical analysis required to be more definitive of the times taken. This testing will be restricted to the Insertion, Shell and Quick Sorts, and will be carried out on lists of 10 items and longer. I will report on this work in some later issue of the magazine.

Use these routines in your own programs. Note that the lines *must* be kept as shown or else the routines will not work. Check that your program does not contain the variable names used exclusively by the routines (or else change the ones in the routines) and ensure that the names of the variables being sorted agree with the ones that you have used in your program. The routines may be extended to cover two dimensional arrays by swapping a complete row of the array, rather than the single element shown above.

Good luck and speedy sorting.

(continued from Page 25)

suggest you cautiously apply only the following changes:

a) Setup the INITIAL COMMAND BUFFER

When prompted by

<*Initial command buffer empty; Is this correct (Y/N):>
reply N

The prompt

<Enter new Initial command buffer:>

will indicate it is ready for you. You should now enter <dir^M>

This will cause CP/M to give you a Directory automatically upon entering the CP/M operating system.

b) Setup the SIGN-ON STRING

Essentially, this is prompted the same as for the L.C. Buffer.

Enter <^@jj^a@^ljCP/M 2.2 Amstrad ...etc>

This will reset the colours to:

Pen - Black (@@=0)

Paper - Cyan (jj=10)

Border - Cyan (jj=10)

Don't forget to answer <Yes> to all other prompts of <Is this correct (Y/N):> and answer <Y> to the request <Do you want to update your system disc (Y/N):>

Remember all CP/M discs are system discs - this costs you 11K per disc! System disc refers to the way the disc was last FORMATTed. See section 3.5 (logging in a disc) of the DDI-1 User Instructions.

Upon successful completion of this task, all future executions of the FORMAT.COM utility from this disc will result in the new setup being carried across.

(continued from Page 23)

```

4))*(1-ABS(SGN(m-2))) : l=0: y1
=y: IF m>2 THEN 350 ELSE m=m+1
2: y1=y-1
350 m=m+1: z=INT (365.25*y1)+INT (
30.6*m)-1: z=z-7*INT (z/7): m=m
1
360 q$(1)=STRING$(2*z,32)+q$: q$(1
)=MID$(q$(1),1,14): l=7-z: l=0=1
: ' MOVE IN FIRST LINE OF MONTH
370 l=10+1: a=(1+7+d-ABS(1+7-d))/
2: q$(10)=MID$(q$,2*1+1,2*(a-1
)): l=1+7: IF l<d THEN 370
380 ' PRINT MONTH
390 CLS: LOCATE 2,4: PRINT e1$: PRIN
T TAB(2);"*": TAB(40-<LEN(m$<
m>+10>)/2); m$(m);",": y: TAB(3
8);"*": PRINT TAB(2);e1$: PRINT
TAB(2);"* S M T W
T F S *": PRINT TAB(
2);e1$
400 x=2: t=8: a1=3
410 FOR l=1 TO 6: LOCATE x,t+1: PRI
NT "*";
420 FOR l1=1 TO 7: LOCATE x+a1,t+1
: PRINT MID$(q$(1),11*(2-1,2):
a1=a1+5: NEXT l1: PRINT TAB(38)
;"*";
430 a1=3: NEXT l
440 PRINT TAB(2);e1$
500 z$=UPPER$(INKEY$): IF z$="." T
HEN 630
510 IF z$="," THEN 640
520 IF z$="R" THEN 80
530 IF z$="1" THEN m1=1: GOTO 300
540 IF z$="2" THEN m1=2: GOTO 300
550 IF z$="3" THEN m1=3: GOTO 300
560 IF z$="4" THEN m1=4: GOTO 300
570 IF z$="5" THEN m1=5: GOTO 300
580 IF z$="6" THEN m1=6: GOTO 300
590 IF z$="7" THEN m1=7: GOTO 300
600 IF z$="8" THEN m1=8: GOTO 300
610 IF z$="9" THEN m1=9: GOTO 300
611 IF z$="O" THEN m1=10: GOTO 300
612 IF z$="N" THEN m1=11: GOTO 300
613 IF z$="D" THEN m1=12: GOTO 300
614 IF z$=";" THEN y=y+1: GOTO 300
615 IF z$="-" THEN y=y-1: GOTO 300
620 GOTO 500
630 m1=m1+1: IF m1>12 THEN m1=1: y=
y+1: GOTO 300
635 GOTO 300
640 m1=m1-1: IF m1<1 THEN m1=12: y=
y-1: GOTO 300
650 GOTO 300

```