VICTORIA 99'er GROUP NEWSLETTER
APRIL 1983

****

Editor's Comment:

I have a few announcements to make this month. First off, this newsletter will carry classified ads as of next month. The ads will cost 10 cents a word, but will be restricted to user group members until legal details are investigated.

Included with this newsletter is a questionaire that we would like every member to fill out. This information will be indexed, so any member can check to see if another member has a piece of hardware or software he or she is interested in seeing. This gives you the opportunity to see an item at work before you buy it.

I would also like to remind you that we need articles for this newsletter. They can cover any subject related to micro-computers in general, and the TI-99/4A in specific. I would also like to ask for volunteers for the editorial staff of this newsletter. Any person with articles or who would like to help in the publication of the newsletter can contact me at 383-2394.

Evan Smith
Editor.

****

PROGRAM CONTEST

We are happy to announce our first programming contest. The rules are as follows:

1) Entrants must be amateur programmers (no programmers with formal post-secondary training).
2) Programs can be on any topic, and must be written in TI Basic or TI Extended Basic.
3) Entry fee will be $2.00.
4) Deadline is the June 1983 general meeting.
5) Bring entry fee, and program listing on cassette.

First Prize: One half of the Entry fees.
Second Prize: One quarter of the Entry fees.

Decision of the Judges is final.
Contest is open to all members of the Victoria 99'er Users Group and the Nanaimo 99'er Users Group.

****

FOR NEW MEMBERS:

There is a library of programs available on cassette. The newest one being added is one written by Johan Van Imschoot which is a character design and save to tape program. If you've written a program(s) and it seems good to you, why not share it with the club and donate a copy. This way, we can all share and learn at the same time.

MEDIUM RESOLUTION PLOTTING PROGRAM
(Extended Basic)
by Rob Sorensen

LISTING:

```
10000 SUB PLOT(Y,X)
10005 DIM TAB$(15)
10010 IF A < > 0 THEN 10090
10020 FOR A= 128 TO 141 ::READ A$ :: CALL CHAR(A,A$) :: NEXT A
10030 DATA FOFOFOF,OFOFOFOF,OOOOOOOOFOFOFOF,OOOOOOOOFOFOFOF,
FFFFFFFF,OOOOOOOOFFFFFFFF,OFOFOFOFFOFOFOF,OFOFOFOFOFOFOFOF,
10040 DATA FOFOFOFOOFOFOFOF,OFOFOFOFFOFOFOF,OFOFOFOFFFFFFFFF,
FOFOFOFOFFFFFFFF,FFFFFFFFFOFOFOF
10050 DATA FFFFFFFFOFOFOFOF,FFFFFFFFFFFFFFFF
10060 FOR A=1 TO 15::READ TAB$(A)::NEXT A
10070 DATA 128132134136,132129137135,134137130133,136135133131,
132132140141,139138133133,134140134139,141135138135
10080 DATA 136141139136,140137137138,142138138138,139142139139,
140140140142,141141142141,128129131131
10090 XX=(X+1)/2 :: YY=(Y+1)/2 :: CALL GCHAR(INT(YY),INT(XX),CH)
10100 IF CH=142 THEN SUBEXIT
10110 IF (CH > 127)*(CH < 142) THEN TY=CH-127 ELSE TY=15
10120 IF (YY=INT(YY))*(XX=INT(XX)) THEN A=1 ELSE IF YY=INT(YY) THEN
A=4 ELSE IF XX=INT(XX) THEN A=7 ELSE A=10
10130 PCH=VAL(SEG$(TAB$(TY),A,3))::CALL HCHAR(INT(YY),INT(XX),PCH)::SUBEND
```

To use the above subroutine, type: CALL PLOT(Y,X)
where 1 <= Y <= 48 and 1 <= X <= 64

Note: Character sets 13 and 14 are used by the subprogram so the user cannot redefine those characters. You can add color by the following statement:

```
10015 CALL COLOR(13,F,B,14,F,B)
```

where F=foreground color and B=background color

Whenever you write a program that needs medium-resolution plotting, load this program and type your program in ahead of it.

****

LOGO ARTICLE
by Johan Van Imschoot

(A demonstration of TI LOGO was promised for the April 83 meeting.) TI LOGO is probably the LOGO requiring the least expensive system to implement it. Unfortunately, it still does require extra memory and is therefore not usable by many TI 99/4A owners.

For those who have the extra memory, or who are planning to extend their system, I highly recommend LOGO. There are a number of reasons for this, which I hope to enumerate below. I will not address advantages or disadvantages of details of TI LOGO, but will address myself to the more general features.

LOGO lends itself to logical, structured program desgn and eliminates unnecessary mechanics such as Basic's line numbers (which indeed do not exist in most programming languages).

LOGO sub programs (procedures) can be written, tested, and saved independently, making program development much easier than it is with TI or even TI EXTENDED BASIC.

The independent testing point is extremely important. It derives from the fact that the LOGO environment is extremely interactive. Whatever commands are entered take effect immediately. Unlike Basic, these commands are not a limited set: they include all user-defined procedures.

Version II of TI LOGO does apparently include music ability, but commands will be more musical than the basics', which use numbers. As far as I know at the time of writing, there is no feature for accessing the speech synthesizer.

Like the Basic, LOGO can access RS232 (like for a printer, disk and tape). (RS232 in version II only)

Graphics is a LOGO strong point. The 4's standard 16 colours are available. There are four graphics modes or features which facilitate graphics of various types. One is the familiar character-based graphics of the TI 4 console BASIC. Another is the sprite graphics familiar from EXTENDED BASIC (except there are 32 sprites, 4 more, and they are interactively usable). There are two additional graphic abilities: turtle mode and dot plotting. The LOGO turtle is probably the most famous LOGO feature and the one which probably most appeals to kids.

Drawing with the turtle (which is simply a directional eraser/pen) tends to be easy because it can be related to how one imagines moving: as occuring from where one is to where one goes, simply by turning and moving. This is easier to describe and grasp than movement using cartestian co-ordinates, as with dot-plotting. The latter is however available and could probably be used for plotting graphs, for example. Turtle drawing lends itself readily to short descriptions for complex drawings that contain pattern repetition, by using recursion.

Recursion is another feature of LOGO which is appealing. Although in general recursion is not required for programming, it does make solving certain types of problems much easier. (For the uninitiated, and simplistically, recursion in programming exists when a procedure directly or indirectly calls itself.)

LOGO is a list-processing type language, descendant of LISP and can be used for symbolic processing, and functional style programming, yet has enough conventional imperative programming features that a newcomer would not need to learn a new programming style.

If a command does not exist in LOGO, then you create it. This is so because the procedure which would implement it is called by name only, and because it is independent, and interactive. Unfortunately, this does not mean that anything is possible in LOGO. It is of course necessary to build up new commands from existing ones, and these 'primitives' will not allow complete control over the machine. Nor unfortunately, as far as I know (though version 2 may change this), is there any access to assembly language from LOGO.

Another drawback of TI LOGO is that floating point is not available. One can only work with integers, and would have to create one's own decimal numbers, which would only slow things down. Therefore, for programming mathematical computations, choose BASIC over LOGO.

On the other hand, for teaching programming, choose LOGO. It is easier to learn (in fact, it is used by elementary school children), but perhaps more important, tends to encourage good programming style (top down design of problems, using functions, identifying data structures--one doesn't need to know this terminology to actually be doing it). Basic, on the other hand, especially many traditional basics, such as the TI 4 console basic, can encourage sloppy style (sometimes known as spaghetti code).

The fact that LOGO has been designed to be easy to use and learn (for example, use a colour name rather than a number, or another example, design interactively a character, rather than use hexadecimal descriptions), this does not mean that it is a language for kids only. It is a complete and powerful list-processing language that even programmers would enjoy using.

****

SORTING SUBROUTINES

Here are two versions of the Shell Sort algorithm in TI Basic and TI Extended Basic.

```
100 REM TI BASIC SHELL SORT
110 DIM A(50)
115 INPUT "NUMBER OF ITEMS TO BE SORTED? ":SI
120 FOR I=1 TO SI
130 RANDOMIZE
140 A(I)= INT(RND*100+1)
150 PRINT A(I)
160 NEXT I
170 PRINT ::
190 REM SORT
200 B=1
210 B=2*B
220 IF B <=SI THEN 210
230 B=INT(B/2)
240 IF B=0 THEN 500
250 FOR I=1 TO SI-B
260 C=I
270 D=C+B
280 IF A(C) <= A(D) THEN 340
290 AA=A(C)
300 A(C)=A(D)
310 A(D)=AA
320 C=C-B
330 IF C > 0 THEN 270
340 NEXT I
350 GOTO 230
500 REM  PRINT SORTED ARRAY
510 FOR I=1 TO SI
520 PRINT A(I)
530 NEXT I
540 END
```

* *

```
100 REM  TI EXTENDED BASIC SHELL SORT
105 OPTION BASE 1
110 DIM A(50)
115 INPUT " NUMBER OF ELEMENTS? ":SI
120 FOR I=1 TO SI::RANDOMIZE::A(I)=INT(RND*100+1)::PRINT A(I); :: NEXT I ::
    PRINT :  :
```

```
130 CALL SORT(SI,A())
140 REM  PRINT SORTED ARRAY
150 FOR I=1 TO SI::PRINT A(I); :: NEXT I
160 END
1000  REM   SORT ARRAY
1010 SUB SORT(SI,A())
1020 B=1
1030 B=2*B :: IF B <= SI THEN 1030
1040 B=INT(B/2) :: IF B=0 THEN SUBEXIT
1050 FOR I=1 to SI-B :: C=I
1060 D=C+B :: IF A(C) <= A(D) THEN 1080
1070 AA=A(C) :: A(C)=A(D) :: A(D)=AA :: C=C-B :: IF C > 0 THEN 1060
1080 NEXT I :: GOTO 1040
1090 SUBEND
```

* *

CAUTION: The array A( ) must be dimensioned to a larger or equal size than the number of elements to be sorted.