THE GUILFORD 99'ER NEWSLETTER

VOL.5 NO.1 JANUARY 1988

The Guilford 99'er Users' Group Newsletter is free to dues paying members (One copy per family, please). Dues are $12.00 per family, per year. Send check to 3202 Canterbury Dr., Greensboro, NC 27408. The Software Library is for dues paying members only. (George von Seth, Editor)

OUR NEXT MEETING

DATE: JANUARY 5, 1988. TIME: 7:30 PM PLACE: Glenwood Recreation Center 2010 S. Chapman Street.

Topic of the meeting this month will be trouble-shooting the TI system. Bring your problems with you as there will be various demonstrations using the diagnostic programs which are available in our library.
Again, a reminder that 1988 DUES ARE NOW DUE AND PAYABLE!!!!

PEEKS AND POKES

The meeting was a good time for everyone attending, and we missed you who could not make it for one reason or another. Thanks to all the Wives who brought in the goodies, thanks Girls! Larry Sponn gave a nice "Night before Christmas" demo. As you know, Larry is trying to get more reconition for Logo, so naturally, his program was in Logo II.

I would like to thank all the members for "putting-up" with me for my term as President. It was a very nice experience. I just hope I can do a good job for you as your Secretary. I will try to attend all meetings and also keep the minutes for you. Hopefully, we can print the minutes in the newsletter so those of you from out of town and the ones who seem to forget the meeting night can keep up with what is going on. I will also try to keep all newsletters from our other 99er friends in a folder so that you can check them out and read them. There is a ton of good information coming in all the time. I would also like to give a personal thanks to my staff that served with me. Thanks to Bob Carmany, Mike Garrett, and Herman Geschwind for the fine job they have done for the Club. Also, I would like to thank Carl Foster and Mike Garrett for their help in getting the newsletter printed for us. Thanks to all you guys.

I want to give a warm welcome to Walter Tietjen and Scott Hughes who are our two newest members. Walt is living in Raleigh and Scott's home is Burlington. Welcome fellows. I asked Emmet if he and Scott were related, and he said, "Who knows, we could be"!! I hope all three of our members down that way can get together and car pool.

THE WOES OF MAIL ORDER

Around the first of September, Herman let me borrow his 1200 baud smart modem as I was having trouble with my Bell/212. Needless to say, the auto dial had me spoiled from the first use of it. So when my lovely wife asked what I would like as a Christmas gift, I told her of the Avatex that I had just been reading about in Computer Shopper, and it was only $85. dollars. I didn't think she would go that much, but much to my surprise, she told me to write all the ordering info and she would call next day. That was around the second week in September. After about two weeks, I received from UPS my modem from Megatronics. I told the wife that I just had to try it to see if it was ok so I wouldn't have to wait at Christmas if anything was wrong. Well it was!! After about 10 minutes, the DT led started blinking and the computer locked-up. I disconnected everything from the computer's RS232 card, and had only the AC cord hooked up. Sure enough, after 10 minutes, the led started blinking, and it came on steady. I also noted that in the rear there was quite a bit of heat build-up. The next day, I called Megatronics and informed the lady of my problem. She must have been familiar with similar happenings, for she asked me right off if there was any heat at the rear of the modem. She asked me to call back after 4:00 that afternoon and talk to a gentleman about returning it. I got the man when I called back that evening and after explaining my problem, he gave me a retrun number for the modem and advised me to send it back at my expense. I went next day to UPS and forked out another $4.13 to send it back. I had planned a two week fishing trip to the beach with a friend the next week, so I hoped that when I returned, my modem would be here. No such Luck. It is now the first week of December and no modem. Can you imagine what this would have been like if I had waited until Christmas to open it!!?

I guess I have no reason to blame Megatronics, as they had no idea it was bad. All I am trying to say is, sometimes the cheapest way to go is not the best way to go. I am sure that if I had shopped all over Greensboro, or even gone to Charlotte, I could have found the same thing and not have gone through all this bother, but I trusted my luck to mail order this last time, and I mean last time!!

I hope your Christmas will be full of Joy and Wonder and as we all enjoy the happiness of having all the family together and welcoming old friends, try and keep the thought of what happened so many years ago that caused this world celebration. Until we meet again, have a very Merry Christmas, and a very,very happy and prosperous New Year, and keep that TI hacking.! (Submitted by L.F. "Mack" Jones)

FORTH FORUM

This month, I'm going to turn the column over to one of the truly talented Forth programmers around --- Mr. Lutz Winkler. Lutz's TI-Forth tutorials are and excellent way to start learning Forth. This columns, however, deals with the conversion of TI-Forth screens to D/V 80 files (and vice-versa). Without further delay, here is Lutz's column:

"Some time ago, there was a program floating about to convert Forth screens to DIS/VAR 80 files. If memory serves me correctly, it was 13(!) screens long and required a lot of typing by the user including that ugly F-D" word. Now, in the September issue of the LA 'TopIcs' newsletter (with credit to the TINS newsletter) I spotted a three-screener to do the same thing. Though 16-line screens are shown, the article states that the code is Wycove and I/O specific words would have to be replaced to make it work in TI-Forth. If you don't know what Wycove's I/O words mean (or do), this is a bit hard to do. Here, for all the writers who wish to include Forth screens in text files but have no desire to enter them manually, is a two-screener that not only transfers screens to D/V 80 files, but also does it in reverse. And it works in standard TI-Forth:

```
( SCREENS>FILE ) BASE->R DECIMAL 0 CLOAD F>S 68 CLOAD STAT
 HEX 0 VARIABLE BUF 4E ALLOT
 PABS 2+ BUF 1400 FILE XF XF
: FNAME ( -- cnt ) CR ." Filename: " PAD 40 20 FILL
       PAD 1+ 30 OVER OVER EXPECT -TRAILING 2- -DUP ;
:SETFILE ( n -- ) IF XF SET-PAB VRBL 50 REC-LEN DUP
                   ROT 1- C! 1+ PAD PAB-ADDR 9 + ROT VMBW
                  ELSE DROP APPND THEN ;
: S>F FNAME SETFILE OPN
      CR ." Start from screen: " QUERY INTERPRET
       CR ." End with screen: " QUERY INTERPRET 1+ SWAP
       DO I BLOCK 400 OVER + SWAP
        DO I BUF 20 MOVE BUF 40 -TRAILING 1 MAX WRT DROP
       40 +LOOP LOOP CLSE
       CR CR ." S>F, F>S or FORGET BUF" CR ;
R->BASE

( FILE>SCREENS )
BASE->R HEX
: EOF STAT 1 AND 0= ;
: F>S FNAME SETFILE OPN
       CR ." Destination screen: " QUERY INTERPRET
       CR ." How many screens?: " QUERY INTERPRET
       OVER + SWAP
   DO EOF IF I BLOCK UPDATE 400 OVER OVER 20 FILL OVER + SWAP
    DO EOF IF RD BUF I ROT CMOVE ELSE LEAVE THEN
    40 +LOOP
```

```
10         THEN
11    LOOP CLSE FLUSH
12    CR CR ." F>S, S>F or FORGET BUF" CR ;
13 R->BASE
14 CLS CR ." Enter S>F for screens to DV80 file,"
15     CR ." or F>S for DV80 file to screens." CR CR ;S
```

The original version of this routine comes from Michal Jaegermann of the Edmonton, Alberta user group and is only one screen long. However, it works only with the enhancements he has incorporated in his file I/O load option. I adapted it to work with the normal -FILE load option and added some on-screen prompts which increased its size to two screens. Even though this is a step backwards - at least from Michal's version - it is still a considerable improvement over the 13-screener and a useful one if you don't happen to have Wycove. With the prompts it is self-explanatory and requires only one comment: If you have written a number of (consecutive) screens to a file and wish to append more to the same file, press <Enter> instead of entering a new file name. For example, you have established a file named DSK2.SCREENS and transferred screens 39 through 45 to it. If you opt for S>F again and press <Enter> at the 'Filename:' prompt, the next entries, say from screen 60 to ending screen 66, are appended to the earlier transfer. Your 'SCREENS' file will contain screens 39-45 and 60-66.

Please note that only the contents of each screen is written to the file. For use in a article, such as this one, you will have to insert some blank lines to seperate the screens. For eye appeal you may also want to add line numbers as I have done here."

This marks the last column in this series for a short while. I am going to take a break for several months both to gather material and do some Forth programming for future articles . An expanded version of the 'ATRAX TRACKS" column will replace 'THE FORTH FORUM' for awhile.

BASIC CORNER

V. XB STYLE WITH SUBPROGRAMS

Let's now stand back a bit and look at the best way to construct XB edifices. Assume at this stage that we are in the process of developing a program, but not yet to the point where scrunching program length has become important. The first thing to note is that by giving the subprograms good descriptive names you have already gone a long way to making your program self-explanatory.

How big should individual subprograms be allowed to get ? After all, one of the reasons for using them is to break up big programs into manageable hunks. We will use the term 'line' to refer to a multi-statement XB line identified by a line number. My own prejudice is that, except in special circumstances, subprograms should be no more than about 10 lines long, and mostly rather less than that. What makes an exceptional circumstance? An obvious one is in title blocks, like that in SIMPLIST which was left as an almost bare stub. A full version would provide graphics and advice screens, which can be tediously long to write, but contain very little in the way of branching decisions or variable assignments. Another example is where a familiar routine, that already works, is used with little variation as in COLIST where the disk directory routine from the Disk Manual is incorporated as a subprogram with only minor changes. In any such situation where long subprograms are justified, the lists of parameters passed will be short or non-existent.

The other extreme is short one or two liners which are frequently CALLed for small special tasks, more or less your own customized extension of the built-in set of subprograms. In the middle there are middle-length subprograms with extensive parameter lists and the logic which carries the burden of program flow.

Some subprograms may be CALLed only once from within another subprogram but are of value in making your code easier to read and modify. These are associated with the branching of program flow by means of IF..THEN..ELSE statements. In either TI BASIC or XB, FOR-NEXT loops may extend indefinitely with NEXT acting as delimiter. Unfortunately in extending BASIC to XB, TI did not provide an "ENDIF" statement as in TI-FORTH, but only the 'endif' implied by the end of a XB line. This means that any alternative actions determined by the IF.. condition have to fit within that XB line or involve a GOTO somewhere else unless the usual simple drop-through to the next line is enough. The XB manual already explicitly forbids inclusion of FOR..NEXT loops within IF..THEN..ELSE statements. No doubt you are already used to getting around this little deficiency by placing the looping code in a subroutine and using a GOSUB. Subprograms can be used instead, following THEN and ELSE to give more complex alternative possibilities, but still staying within the confines of a single line with a minimum of leaping about with GOTOs.

This brings us to the subject of the 'dreaded GOTO'. A great deal of heat, and not necessarily much light, has been expended on this subject. It is after all just another statement available in many languages, and has perfectly predictable immediate consequences. At the machine code level, jumps enable the computer to do more than just chomp along a single track of instructions. The question is whether it is help or hindrance in high level languages, and whether other ways of controlling program flow can replace its explicit use to advantage. TI-FORTH does without it, but that most procedural of languages, TI-LOGO, still finds it useful. Pascal tries to do without it. What we do have is XB, and XB can't do without GOTOs. If anything should be considered as reprehensible in a high-level language, it is any need to provide PEEK and POKE.

The great weakness of GOTO as a language element is that it is so readily abused, because undisciplined use makes the program code inefficient and hard for people to follow. The genuine message from 'structured programming' ideas is not that BASIC is bad for having GOTOs, but that most BASICs (TI console Basic is typical) make it necessary for the programmer to exercise real restraint if terrible tangles of GOTOs are to be avoided.

Once you use XB subprograms to chop up a program into small hunks, then you have automatically eliminated great leaps around with GOTOs. All you need then is to remember the comments on using subprogram CALLs as statements in IF..THEN..ELSE and take a little care in laying out the logic flow, you will find it very much easier to debug or develop programs. Backwards GOTOs over more than one or two lines of code, or any forward GOTOs at all, should only occur under the most regular of logical layouts, as in SUB BASICLINE in the SIMPLIST example. Single recursive lines such as in line 620 of SIMPLIST are very effective. It's a pity that the designers of XB didn't add the "MYSELF" function as in TI-FORTH to enhance such constructions.

One last little matter before we go on to other topics. Many languages with local procedures also allow specification of global variables, accessible from any part of the program. XB does not allow for separate global variables, and it can be quite tiresome when a parameter defined at the end of one subprogram chain is only needed at the end of another chain, and has to be passed all the way up and down in parameter lists. A way around this is to use the static value feature of XB subprograms.

```
3000 SUB PAGELENGTH(A,B):: IF A THEN C=B ELSE B=C
3010 SUBEND
```

If the write flag is set as CALL PAGELENGTH(1,66) the value 66 is stored in the subprogram local variable C, while CALL PAGELENGTH(0,PL) will retrieve that value into PL. This is clumsier than having global variables, but is also more protected from unwanted interference. XB does not enforce any hierarchy of subprogram levels, so PAGELENGTH can be written to, or read from, at any level in the program. The example is for one parameter only, but is easily extended.

VI. PRE-SCAN SWITCH COMMANDS

The little supplementary booklet that comes with the current Version 110 of Extended Basic introduces a new pair of reserved words, !@P+ and !@P-. These have the form of a tail remark (XB manual p38) and so are ignored entirely by the earlier V.100 of XB. If the XB interpreter finds an exclamation mark ! outside any DATA string or string enclosed by quotes, it treats the rest of that line as though it were a REM statement. The V.110 interpreter has the added ability to recognize this pair of words beginning with ! as being distinct from normal tail remarks when used as a single word statement. Their use is allowed only at the end of a line so that V.100 just ignores them, not creating any incompatibility problems between versions, something that TI was always conscientious about. TI then couldn't let these commands actually do anything! So why are they there ?

The XB manual addendum, p7, tells the story. These switch commands allow you to control the operation of the pre-scan through the program by the interpreter -- that agonizing time interval after RUN is entered before the program starts executing. The interpreter is grinding its way through your program, byte by byte, ignoring only the messages in DATA, REMs and tail remarks. Other than these there is nothing that it can afford to ignore until it has actually looked at it. The pre-scan sets up the storage areas and lookup procedures for variables, arrays, data, sub-programs and DEFs used by the interpreter as the program runs. Of course once it has set aside space for a variable and its lookup linkages, then it doesn't need to do it again or even to have to decide it has already fixed it up earlier. The pre-scan switch commands allow the programmer, from a superior vantage point, to turn the pre-scan off and on throughout the program so that it only looks at what it really needs to look at to do its job.

What does the programmer gain by going to all this extra trouble? The most obvious result is a reduction of pre-scan time. This can be significant in long programs. The 6 to 7 seconds for TXB, a 12K program, may still seem long but beats 4 times that. In a later chapter we will see how it can be used to fine tune run time behaviour as well. What price does the programmer pay for these benefits? The necessary penalty is the memory space taken by the extra statements. The hidden penalty, incurred while writing programs, is the inscrutable bugs that may be introduced into the code and the loss of some program checking during pre-scan such as FOR-NEXT nesting.

Let's work our way through the XB manual's prescriptions. Some of these help give insight into the way XB conducts its affairs. My experience is that some of the restrictions need not be followed strictly as laid down, as long as the essential spirit is observed, while some are absolute, and others are in between. These last are the ones where it is possible to imagine another version of XB doing things differently while still being according to the book. This is always the danger in using unspecified properties or "undocumented features". It is not such a problem with XB since TI pulled the plug on the 99/4a and made XB a language as dead as Latin.

(1) DATA statements :-

The pre-scan locates the first DATA statement and sets XB's data pointer for the first READ operation to use. If the first DATA is skipped in the pre-scan, then RESTORE must be invoked before the first READ to set the data pointer correctly. If this is done, the XB manual's advice can be ignored.

(2) Variables :-

Each variable must be scanned once, otherwise XB won't have it in its linked list of pointers to names and storage locations. This can be the source of some truly evil program bugs, where a syntax error message results from a line of code which looks perfectly correct. The reason can be that injudicious positioning of pre-scan switch commands has left the interpreter with something that should be a variable, but can't be located as such. Being a non-variable is a much worse fate than merely being set to zero.

OPTION BASE 1 affects how storage is allocated and normally precedes any array references. If hidden from the pre-scan by !@P- then the default 0 will apply.

The manual says that the first occurrence of any variable or array must be included in the pre-scan. This would seem to be necessary for arrays, in the DIM statement, unless you are using the default (no DIM) dimensioning. Simple variables can be pre-scanned anywhere as long as it's at least once. Try the little sample program

```
100 CALL CLEAR :: !@P-
200 I=1 :: PRINT I
300 !@P+
400 I=2
```

Run this program and there will be no problems. Delete line 400 and see what happens. Now you will have a syntax error in a line that by itself is perfectly correct.

(3) Sub-programs :-

The XB manual recommends that the first CALL to any sub-program be included in the pre-scan. It would appear that if the first CALL to a user defined sub-program occurs after its own SUB (from within a later sub-program) then the necessary inclusion of the SUB and SUBEND markers suffices.

Built-in sub-programs of course do not have associated SUB statements, so a CALL must be included in the pre-scan if the program is to run normally. Try this example.

```
100 FOR I=1 TO 1000 :: !@P-
200 CALL SCREEN(12)
300 !@P+
400 NEXT I
500 SUB ANYTHING :: CALL SCREEN(3):: SUBEND
```

This will run even though SCREEN is pre-scanned only in a subprogram. Delete line #500 and it will crash if you are running XB with the 32K memory expansion. In VDP RAM (console only) it still executes but only at about 1/3 the speed.

What happens if an array is referenced in the parameter list of a sub-program, but not dimensioned until a later sub-program? If you recall the discussion on passing arrays by reference, you won't be surprised to find that XB is smart enough to hold over assigning space for the array until it comes across a genuine program reference. Try this little example

```
100 CALL SECOND
200 SUB FIRST(A()):: PRINT A(20):: SUBEND
300 SUB SECOND :: !@P-
400 DIM A(20):: CALL FIRST(A())
500 !@P+
600 SUBEND
```

This program crashes with a syntax error in 400 in SECOND. Now delete the pre-scan commands and the program will run. If you further delete DIM A(20):: in line 400 the program will crash in 200 with a subscript error.

(4) DEF,SUB and SUBEND :-

Do as the book says. XB needs these in the pre-scan to set things up correctly.

The pre-scan switch doesn't have much effect unless the program is of substantial size, so it isn't worth worrying about too much in the early stages of a program's development beyond being prepared for the possibility. The XB manual supplement (p10) shows how all variable and sub-program declarations may be gathered together to minimize the range of the pre-scan, by using a GOTO to jump over the list to the first executable statement. This can be gotten away with since XB does not do a complete check for correct syntax until it comes to execute the line. This is the only virtue one can ascribe to XB's failure to reject all invalid lines at entry time. The same technique can be used within a sub-program, and I have found it very convenient for this same GOTO to reserve a hiding place in which to tuck away the subroutines accessed by GOSUBs within the sub-program.

SPIDERS ETC.

Now funnelweb spiders aren't exactly the nicest critters to be found around Funnelweb Farm, but they do have the virtue that if you don't bother them they leave you alone too. Unfortunately the bugs that infest Extended Basic aren't nearly so accomodating in keeping out of the road in the first place, and insist on making their presence felt.

I have looked at a few in previous XB tutorials where they came up naturally in the subject matter, mostly in ACCEPT AT. From now on Entomology Corner will look at XB bugs as they come to light. This time we have two beautiful specimens. The first of these was exposed in the Spring 85 issue of TI*MES from the UK by John Bingham. To see it at work, enter the following little program

```
100 I=1 :: IF I=1 THEN J=1 :
: GOSUB 200 ELSE K=1 :: J=2
110 PRINT K;J :: STOP
200 RETURN
```

Before you run this, predict what it is going to print out ! Then run it and see what you get. Next reverse the order of the statements between THEN and ELSE and run it again. Now it should work the way you expected, K=0 and J=1. If your XB gets it right the first time, it's different from the one I have. The presence of the GOSUB just before ELSE seems to have upset XB's mechanism for keeping track of ELSE, and the program has gone ahead and ignored the ELSE and the statement after it and executed the following statement which it should have ignored entirely while proceeding to the next line. Try substituting a dummy SUBprogram CALL for the empty GOSUB. XB then works just as expected. Yet another reason for using SUBprograms instead of GOSUBs.

The XB manual lays down a few prohibitions on what can go into IF doesn't mention this little beauty. It does seem, despite the warnings, that FOR..NEXT loops can follow the final ELSE without problems, but this usage is not to be recommended as it may not hold good for all XB modules.

I must admit that reading this news had me a little worried, as I have written long and thoroughly debugged XB programs with some tricky IF..THEN..ELSE footwork, and had never picked up this problem. How come ? The first saving grace is that the Tutorial advice I gave is for real, and I use very few GOSUBs and very many SUBprograms unless I am absolutely desperate for more bytes. This was frequently the case in the writing of TXB, and the central SUBprogram, one of 12 in the program, itself contains 12 subroutines written in to save bytes. Careful study of the code for TXB showed that none of the GOSUBs was written in a way that would let this evil bug loose. When you look back at something like this, you wonder whether you had scrapped particular pieces of code that never quite worked properly, for entirely wrong reasons.

The second bug mentioned has not yet been fully explored. It showed up in a CALL LINK from XB to an assembler routine which also passed in a string variable to be examined by the routine. It happened in the COLIST program, the all-singing all-dancing version of the SIMPLIST program which appeared in a XB Tutorial in those earlier days in SND. The symptoms were that the machine crashed utterly when it was printing out a line of its own listing, after it had already printed several hundred lines, many quite similar to the one that caused to the trouble. Not just an error caught and reported by XB, but a full blown paralytic seizure. It turned out after some TRACE work to be in the very line that was being processed for printing, and to be asociated with the LINK name being at a particular position in the line of text being passed in with STRREF. The problem seems to be CALL LINK extending its link name search over places it shouldn't orter, but more research is needed. Further reports in Entomology Corner in a future issue.

One disappointing discovery came up in the bug hunt. I disassembled the XB machine code utilities loaded by CALL INIT to see if the problem was in STRREF. The good news is that that code looks OK, but the bad news is that STRREF reads strings out of VDP in the same slow way that the console does, 1 byte at a time, resetting the VDP addresses each time. This is the tortuously slow way GPL does it because the console has hardly any CPU RAM, but it scarcely seems necessary in STRREF which can only be used when there is expansion RAM present.

Another bug in CALL LINK has been reported in the US of A in some older XB modules, I suspect prior to the models of V110 sold in Australia. This comes when the link name is supplied as a string variable, as in CALL LINK("A$"). If a garbage collection is performed in VDP RAM by the XBasic interpreter in between assigning A$ and using it in CALL LINK, this routine would lose track of where A$. I have not encountered this bug myself, but I will try to stir it up in the XB modules I have. This reported bug brings to mind the strange state of affairs in XB where it is possible to DELETE a file by string variable reference but not to RUN a program file by similar reference, leading to a smal cottage industry of ways around this deficiency.

While we are in the business of making corrections to all and sundry, there is a minor one to last month's issue. Nothing substantive but only to emphasis of a comment made in passing. That was in respect of finding th DIMensions of an array passed as an argument by CALL LINK. It is defined for E/A Basic in the stack entry built by LINK, but you have to extend this to XB by implication along with a lot of other poorly defined items. The discussion does not tell how this stack entry relates to the internal variable table.

For this issue Entomology Corner will also serve as the Extended Tutorial, and the regular subject matter will be resumed next time. The price you pay for getting the Tutorials written is that you have to put up with my whims of the moment. I have been hard at work on running TI-Writer from XB. An early version is available at meetings or from Funnelweb Farm as a public domain program. Further developments, not yet stabilized enough for general release, include a Formatter smart enough to know the file-name last used in the Editor before switching to the Formatter, and a Show Directory that works from within the Editor, just like the real thing except better.

A significant measure of the worth of a User Group is the range and quality of Public Domain software produced by the members. The HV99 group intends to have a high profile in this area, both in programs released and in original items published in the HV99 News. We are aiming to be one of the groups with a solid and useful output of original material, along with thoroughly evaluated material from other like minded groups. I don't see us reprinting much from TISHUG Newsdigest on its recent form. Glossy covers and two page graphics spreads will rank near the bottom of our list of priorities.

LOGOING

By
Larry Spohn

As I have written in the past, LOGO can do more than just draw colorful pictures on your screen. The following program, which was demonstrated at our last meeting, is an example of LOGO's simple interactive abilities.

The program should now be available in our library. So, it is printed here not to have you re-key it, but to further illustrate the demonstration and to aid those who could not make the last user group session. Also, of course, printing the program makes it available to those who do need to key it in because they are using tape for storage.

The program idea is borrowed from Donna Bearden's excellent book, "A Bit of LOGO Magic." But the application is my own. I wrote two versions: one with kids in mind; the other--a little more sinster--is for "older" kids (the rest of us). This is the kids version.

The main or "calling" program is "TO STORY." So to execute or run it, type STORY. STORY first clears the screen, sets LOGO for non-graphics and then calls just three procedures, first QUESTIONS; second, PAUSE; and third, ANSWERS. This sequence is important for the program to operate correctly. However, let me stress again that outside the calling program, it matters not at all where the other procedures are type in. LOGO finds them whether they come before or after the main calling program. Of course it is easier to understand and follow a program, if it is written sequentially.

Notice that procedure QUESTIONS only purpose is to extract input from the storywritter at the keyboard and store it. LOGO stores the answers (HOLIDAY, NAME, BUILDING, ANIMAL) in memory for recall later by the procedure ANSWERS. It's that simple. Of course, the answers are the programs variables which can be altered each time the program is run. The command MAKE names the variables for calling by procedure ANSWERS. The colon after the bracketed story statements tells LOGO to go find the variable named. ANSWERS merely prints the predetermined story, inserting the appropriate variations (the variables input through QUESTIONS).

The format is tedious, but by necessity because STORY seeks to humourously personalize a very familiar story by alternating print statements of the story with print statements from the story teller's own list of inputs. Note that when a print statement requires both, the command SENTENCE is used. Essentially like a sentence, it strings things like words and phrases together, in this case the prepared story and the variables. SENTENCE is a listing tool.

I can tell you that kids love this. It's fun the first time through; but, once they realize the computer will output whatever they input, the resulting stories can be hillarious, silly, crude, even vulgar. But always fun.

```
-----------

TO ANSWERS
CS
CB :GREEN
PRINT SENTENCE [TWAS THE NIG
HT BEFORE ] :HOLIDAY
PRINT SENTENCE [WHEN ALL THR
OUGH THE ] :BUILDING
PRINT [NOT A CREATURE WAS ST
IRRING ]
PRINT SENTENCE [NOT EVEN A ]
 :ANIMAL
PRINT SENTENCE [THE ] :CLOTH
ING
PRINT [WAS HUNG BY THE CHIMN
EY WITH CARE ]
PRINT SENTENCE [IN HOPES THA
T ] :SOMEONE
PRINT [SOON WOULD BE THERE ]

PRINT SENTENCE [THE ] :FAMIL
Y
PRINT SENTENCE [FAMILY WAS A
LL NESTLED IN THEIR ] :ITEM

PRINT SENTENCE [WHILE VISION
S OF ] :SWEET
PRINT [DANCED IN THEIR HEADS
 ] WAIT 1500
CS
PRINT SENTENCE [AND MAMA IN
HER ] :FEMALE
PRINT SENTENCE [AND I IN MY
] :MALE
PRINT SENTENCE [HAD JUST SET
TLED DOWN FOR A LONG WINTER'
S ] :SOMETHING
PRINT SENTENCE [WHEN OUT ON
THE LAWN THERE AROSE SUCH A
] :NOISE
PRINT SENTENCE [I SPRANG FRO
M MY ] :ITEM
PRINT [TO SEE WHAT WAS THE M
ATTER ]
PRINT SENTENCE [AWAY TO THE
WINDOW I FLEW LIKE A ] :SLOW

PRINT SENTENCE [TORE OPEN TH
E ] :PART
PRINT SENTENCE [AND THREW UP
 THE ] :ANOTHER
WAIT 1500
CS
```

```
PRINT [WHEN WHAT TO MY WONDE
RING EYES SHOULD APPEAR ]
PRINT SENTENCE [BUT A MINATU
RE ] :SLEIGH
PRINT SENTENCE [AND EIGHT TI
NY ] :REINDEER
PRINT SENTENCE [WITH A LITTL
E OLD ] :DRIVER
PRINT [SO LIVELY AND QUICK ]

PRINT SENTENCE [I KNEW IN A
MOMENT IT MUST BE ] :SOMEONE

PRINT SENTENCE [MORE RAPID T
HAN EAGLES HIS ] :REINDEER
PRINT [THEY CAME ]
WAIT 1500
CS
PRINT [AND HE WHISTLED, AND
SHOUTED AND CALLED THEM BY N
AME ]
PRINT SENTENCE [NOW ] :DWARF

PRINT SENTENCE [NOW ] :DWARF
TWO
PRINT SENTENCE [NOW ] :DWARF
THREE
PRINT SENTENCE [NOW ] :DWARF
FOUR
PRINT SENTENCE [NOW ] :DWARF
FIVE
PRINT SENTENCE [NOW ] :DWARF
SIX
PRINT SENTENCE [NOW ] :DWARF
SEVEN
PRINT SENTENCE [NOW ] :DWARF
EIGHT
PRINT [TO THE TOP OF THE POR
CH, TO THE TOP OF THE WALL!
]
PRINT [NOW DASH AWAY, DASH A
WAY, DASH AWAY ALL." ]
WAIT 1500
CS
PRINT SENTENCE [SO UP TO THE
 HOUSE - TOP THE ] :REINDEER

PRINT [THEY FLEW ]
PRINT SENTENCE [WITH A ] :SL
EIGH
PRINT SENTENCE [FULL OF ] :T
OYS
PRINT SENTENCE [AND ] :SOMEO
NE
```

```
PRINT [TOO! ]
PRINT [AS I DREW IN MY HEAD,
 AND WAS TURNING AROUND ]
PRINT SENTENCE [DOWN THE CHI
MNEY ] :SOMEONE
PRINT SENTENCE [CAME WITH A
] :NOISE
PRINT SENTENCE [HE WAS DRESE
D ALL IN ] :FUR
PRINT [FROM HIS HEAD TO HIS
FOOT, ]
PRINT [AND HIS CLOTHES WERE
ALL TARNISHED WITH ASHES AND
 SOOT ]
PRINT SENTENCE [A BUNDLE OF
] :TOYS
PRINT [HE HAD FLUNG ON HIS B
ACK ]
PRINT SENTENCE [AND HE LOOKE
D LIKE A ] :PEDDLER
PRINT [JUST OPENING HIS PACK
 ]
WAIT 1500
CS
PRINT [THE STUMP OF A PIPE H
E HELD TIGHT IN HIS TEETH ]

PRINT SENTENCE [AND THE SMOK
E, IT ENCIRCLED HIS HEAD LIK
E A ] :WREATH
PRINT SENTENCE [HE HAD A ] :
BROAD
PRINT SENTENCE [FACE AND A L
ITTLE ROUND ] :BELLY
PRINT SENTENCE [THAT SHOOK,
WHEN HE LAUGHED LIKE A BOWL
FULL OF ] :JELLY
PRINT SENTENCE [HE WAS A CHU
BBY AND PLUMP, A RIGHT JOLLY
 OLD ] :ELF
PRINT [AND I LAUGHED WHEN I
SAW HIM IN SPITE OF MYSELF ]

PRINT SENTENCE [A WINK OF HI
S ] :BELLY
PRINT SENTENCE [AND A TWIST
OF HIS ] :BELLY
PRINT SENTENCE [SOOM GAVE ME
 TO KNOW I HAD ] :NOTHING
PRINT [TO DREAD ]
WAIT 1500
CS
```

```
PRINT SENTENCE [HE SPOKE NOT
 A WORD, BUT WENT STRAIGHT T
O HIS ] :PEDDLER
PRINT SENTENCE [AND FILLED A
LL THE ] :CLOTHING
PRINT [THEN TURNED WITH A JE
RK ]
PRINT SENTENCE [AND LAYING H
IS FINGER ASIDE OF HIS ] :BE
LLY
PRINT [AND GIVING A NOD, UP
THE CHIMNEY HE ROSE ]
PRINT SENTENCE [HE SPRANG TO
 HIS ] :SLEIGH
PRINT SENTENCE [TO HIS ] :TE
AM
PRINT SENTENCE [GAVE A ] :WH
ISTLE
PRINT SENTENCE [AND AWAY THE
Y ALL FLEW LIKE THE DOWN OF
A ] :THISTLE
PRINT [BUT I HEARD HIM EXCLA
IM AS HE DROVE OUT OF SIGHT
]
PRINT SENTENCE [HAPPY ] :HOL
IDAY
PRINT SENTENCE [TO ALL AND T
O ALL A GOOD ] :NOISE
WAIT 1500
CS
PRINT [BOY, YOU SURE KNOW HO
W TO MESS UP A GREAT STORY!
]
END

TO QUESTIONS
CB :RED
PRINT [NAME ANY HOLIDAY ]
MAKE "HOLIDAY READLINE
PRINT [NAME ANY BUILDING ]
MAKE "BUILDING READLINE
PRINT [NAME ANY ANIMAL ]
MAKE "ANIMAL READLINE
PRINT [NAME ANYTHING PLURAL
TO WEAR ]
MAKE "CLOTHING READLINE
PRINT [NAME SOMEONE FAMOUS ]

MAKE "SOMEONE READLINE
PRINT [WRITE YOUR FAMILY NAM
E ]
MAKE "FAMILY READLINE
PRINT [NAME A FURNITURE ITEM
 ]
```

```
MAKE "ITEM READLINE
PRINT [NAME SOMETHING SWEET
( PLURAL ) ]
MAKE "SWEET READLINE
PRINT [NAME A FEMALE ARTICLE
 OF CLOTHING ]
MAKE "FEMALE READLINE
PRINT [NAME A MALE ARTICLE O
F CLOTHING ]
MAKE "MALE READLINE
PRINT [NAME SOMETHING TO DO
]
MAKE "SOMETHING READLINE
PRINT [NAME A NOISE ]
MAKE "NOISE READLINE
PRINT [WHAT DO YOU CALL SOME
ONE WHO IS SLOW ]
MAKE "SLOW READLINE
PRINT [NAME PART OF BUILDING
 ]
MAKE "PART READLINE
PRINT [NAME ANOTHER PART ]
MAKE "ANOTHER READLINE
PRINT [NAME YOUR FAVORITE KI
ND OF CAR ]
MAKE "SLEIGH READLINE
PRINT [NAME A BEAST OF BURDE
N ( PLURAL ) ]

MAKE "REINDEER READLINE
PRINT [WHAT DO YOU CALL SOME
ONE WHO'S IMBIBBED TO MUCH ]

MAKE "DRIVER READLINE
PRINT [NAME A DWARF ]
MAKE "DWARF READLINE
PRINT [NAME ANOTHER ]
MAKE "DWARFTWO READLINE
PRINT [ANOTHER ]
MAKE "DWARFTHREE READLINE
PRINT [ANOTHER ]
MAKE "DWARFFOUR READLINE
PRINT [ANOTHER ]
MAKE "DWARFFIVE READLINE
PRINT [ONE MORE ]
MAKE "DWARFSIX READLINE
PRINT [I LIED ; GIVE ME ANOT
HER ]
MAKE "DWARFSEVEN READLINE
PRINT [NOW A SMURF ; ANY SMU
RF ]
MAKE "DWARFEIGHT READLINE
PRINT [WRITE YOUR FAVORITE L
IQUOR ]
MAKE "TOYS READLINE
PRINT [NAME AN ANIMAL HIDE ]

MAKE "FUR READLINE
PRINT [WHAT'S YOUR OCCUPATIO
N ]
MAKE "PEDDLER READLINE
PRINT [NAME A GEOMETRIC SHAP
E ]
MAKE "WREATH READLINE
PRINT [WHAT'S YOUR FAVORITE
COLOR ]
MAKE "BROAD READLINE
PRINT [NAME A BODY PART ]
MAKE "BELLY READLINE
PRINT [YOUR FAVORITE DESSERT
 ]
MAKE "JELLY READLINE
PRINT [YOUR FAVORITE ZOO ANI
MAL ]
MAKE "ELF READLINE
PRINT [WRITE ONE OF THESE: S
OMETHING EVERYTHING ]
MAKE "NOTHING READLINE
PRINT [THE MASCOT OF YOUR FA
VORITE COLLEGE TEAM ]
MAKE "TEAM READLINE
PRINT [YOUR FAVORITE MIXED D
RINK ]
MAKE "WHISTLE READLINE

PRINT [YOUR FAVORITE FOUL ]
MAKE "THISTLE READLINE
END
TO STORY
CS
NOTURTLE
QUESTIONS
PAUSE
ANSWERS
END
TO PAUSE
CS
PRINT [ ]
PRINT [ ]
PRINT [CUT ME SOME SLACK ; ]

PRINT [I'M WRITING YOUR VERY
 OWN STORY ]
WAIT 600
PRINT [ ]
PRINT [ ]
PRINT [HERE YOU ARE... ]
END
** DONE **
```