Central Texas

Aug 86

CENTRAL TEXAS 99-4A USERS GROUP THE PAPER PERIPHERAL

From the Presidents Desk

Hello there again people. This month was pretty slow as far as things happening. In fact, the only thing I can think of that happened as far as the group goes, is our illustrious librarian getting ahold of (yes, through another one of his incredible deals) a Cor-Comp Micro Expansion system. Unfortunately, it had this little problem, so we won't see it this month. There was a chance that we would have had it demonstrated, but we were'nt sure whether or not it would be fixed in time, and rather than plan on that as a demo, I decided (we really) to demonstrate PASCAL since I couldn't remember the last time it had been demonstrated, if in fact it had ever been done. So, since I have PASCAL, and have been using it lately, I thought I would show it off. The Micro Expansion system will be shown next month (hopefully). Also, we got a letter from the Herndon House inquiring whether we knew of anyone interested in helping severely handicapped (and not so severely) people learn to use the 99/4A since they had just been donated one. Please let me know if any of you are interested, as I have called them back and said we would help (even if no one else will, I will). By the way, they did mention being able to compensate the person (people) for their time. Oh well, that's all I can think of---wait, one other trivial thing...I am on my involuntary "vacation" right now. Also, I am going to be taking off the end of August to Denver for a couple of weeks, so unless I manage to get things together sooner next month, you won't be seeing my article next month. Oh well bye for now...and... yes... I can say it again... See ya at the meeting.

Joe Pizzi

Programming in C
An Introduction - Part 1

Last month, the Assembly Language Group began a series of classes on the language C. Because there wasn't a lot of advance notice about the class, I thought that I would write an article about the class for the newsletter. If I can, I'll try to write a brief article after every class, this way everyone can get exposure to the subjects. I strongly recommend however, that if you are interested in the language at all, please come to the classes. That way you'll get to ask questions.

The C that we will be using in this class will be a limited version of the full language called Small C. I have a Fairware copy of the diskette if you need it.

Before we get started, I'd like to tell you how I intend to describe elements of the C language in these articles. Most C programs are written entirely in lower case. I'll do this too. However when you see something in these articles that is in upper case, I'll be indicating something of a general nature that goes in a C program at this point. A blank to fill in, so to speak.

To demonstrate (and to start), lets look at the general structure of a C program.

```
    VARIABLE DECLARATIONS
```

```
main()
{VARIABLE DECLARATIONS
 STATEMENTS
}
```

VARIABLE DECLARATIONS indicates where variables are declared in a program. STATEMENTS indicate where the executable statements of the program are placed.

In C, as with many other languages, it is necessary to declare the existance of all the variables in the program before the executable statements. This helps produce faster running programs, as well as helping the programmer to detect mispelled variable names.

In C there are three basic types of variables: Integers (which are 16 bit values), characters (which are 8 bit values), and pointers (which we aren't going to discuss now). One should remember that while integer and character variables are signed (they can be negative), they are the counting numbers and are not floating point numbers. That is, an integer can have the value of 1 or 2, but not 1.5. If you want a program that needs floating point numbers, use BASIC. For most programs, integers are all we really need, and calculating with them can be very fast.

To declare integers in C, use the following statement:

```
int VARIABLE NAME 1, VARIABLE NAME 2, ..., VARIABLE NAME N;
```

Here is an example:

```
int a,b,c;
```

This declares three variables a, b, and c and sets aside memory for them.

Notice that multiple variables can be declared with this statement by separating them with commas. C knows that it has reached the end of the list when it finds the semicolon.

To declare a character, use the following statement:

```
char VARIABLE NAME 1, ..., VARIABLE NAME N;
```

An example would be:

```
char x,y,z;
```

This declares variables x, y, and z and sets aside space for one character for each. That's right, "char" variables can hold only one 8 bit value.

That isn't very useful in most programs. Usually we want to have strings of characters. To do this, we simply declare an array of characters:

```
char VARIABLE NAME [LENGTH];
```

Here are a couple of examples:

```
char w[10];
int d[5];
```

This also shows how to declare an array of integers. The array w has space for 10 characters set up for it, and d has space for 5 integers.

That should be enough about variables for now, lets's move on to statements.

By far the most common statement in a C program is the assignment statement. It looks like assignment statements in most other languages:

```
        VARIABLE NAME = EXPRESSION;
```

Where EXPRESSION is a calculation to be performed. Here is an example:

```
        a = b + 1;
```

In this example, the value of b has 1 added to it, and the result is placed in variable a. Note the semicolon. In C the semicolon is used to indicate the end of all statements.

The operation on the right of the equal sign is called an expression. In C, expressions can be very complex and are very flexible. Some of the operations available are: multiplication (*), integer division (/), addition (+), subtraction (-). Notice the integer division. This means that if you divide 3 by 2, the result is not 1.5 but 1. If you want the remainder of a division, you use a different operation called modulo (%):

```
        b = 5 % 2;
```

Here, the result of 1 is placed in the variable b.

In C, if an expression contains multiple operations, they are performed from the left to the right, with multiplications and divisions being performed first, followed by additions and subtractions. Parenthesis may be used to change this ordering.

Although most people don't think of them this way, C contains some other operations: less than (<), less than or equal (<=), greater than (>), greater than or equal (>=), not equal (!=) and equal (==). No that last one is not a typo. "==" is used to distinguish the comparision operation from "=" that is used for assignment.

When a comparison is made between values using the above operators, the result that is returned is 0 for false and 1 for true. To C, an expression that evaluates to 0 can be considered to be "false", and one that evaluates to anything else is considered to be "true". The importance of this will become more apparent in a few moments.

There are other operators that C supports in expressions, but we'll save those for another class. Next let's turn our attention to the variables and constants of the expressions.

Normal variables are used as you would expect. Just place them in the expression, and the current value of the variable will be used for the calculation. If a character variable is used in an expression with integer variables, then the character variable is converted to a 16 bit value and the calculation proceeds. If an integer expression is assigned to an character variable, then the result of the expression is truncated to 8 bits and placed in the variable.

If we simply wish to use a constant in an expression, that is allowed also. Numeric constants like 1, 2, 100, are treated like decimal integers. If desired, character constants may be used by placing them in single quotes. '0' is a character constant that is the ASCII value of the character 0.

References to arrays are straight forward. Simply use the name of the array and put the subscript in brackets ([]) following it. One dimensional arrays only are allowed in Small C. The first element in the array has subscript 0, and the declaration of the array reserves space for requested number of elements. Thus if 10 elements of the array are reserved, then the valid subscripts range from 0 to 9.

Actually, that is a bit misleading. In C, subscripts are not checked while the program is running. Thus if an array is declared with 10 elements, and the reference is to the 50th element, then C will not stop the program. Instead some memory location past the end of the array will be used.

Another thing that can appear in expressions is a function. This is the name of some other subroutine in the C program. When the name is encountered in the expression, evaluation of the expression is suspended, and the function's code executed. The function's code will generate a value when it is through and this value will be what is used to complete the evaluation of the expression. Functions are a topic in itself, and we'll discuss them in another class too.

Let's turn our attention now to other statements in C, and the "if" statment is a good place to start, it looks like this:

```
        if (EXPRESSION)
           STATEMENT;
```

This says that if the EXPRESSION evaluates to true, then the STATEMENT is executed. Simple enough, lets look at a more complex "if":

```
        if (EXPRESSION)
           STATEMENT-1;
          else
           STATEMENT-2;
```

The "if" statement can also dictate the action to take if the statement is false. In this case STATEMENT-2 will be executed.

Finally, lets look at the "while" statement. This is one way of creating loops in C.

```
        while (EXPRESSION)
           STATEMENT;
```

This says that while the EXPRESSION evaluates to true, the STATEMENT will be executed repeatedly. (The STATEMENT will need to make sure that the expression will eventually evaluate to false.)

Now in the above examples, it was implied that only one statement would be executed, and this is true. However, if you wish to place multiple statements in the "if" or "while" statements, this can be done with the compound statement. This is simply enclosing several statements in braces ({}). Here is an example:

```
        if (a < b)
           {a = b;
            b = b + 1;
           };
```

This should be enough of an introduction to the C language to write a program. Lets take a classic, generating prime numbers between 3 and 100.

```
#asm
 REF PRINTF
#endasm

main()
{int i, j, r;
 i = 3;
 while (i < 100)
    {j = 3;
     r = i % j;
     while (r != 0 & (j * j < i))
       {j = j + 2;
        r = i % j;
       };
     if (r != 0)
        printf("%d is prime\n", i);
     i = i + 2;
    };
}
```

A few notes about this program. First, it's not the most efficent implementation of a prime number generator because that's not what I'm trying to demonstrate here. Second, the "#asm REF #endasm" and "printf" are features of the language that I haven't described yet but they are necessary to let the program output it results. Third, the "&" likewise hasn't be described yet, but in this case it can be read as "and".

In this program, the variables i, j, and r are defined. The variable i contains the current value that we are trying to verify as prime. The variable j cantains the current value that we'll divide into i to see if it will divide evenly and thus prove that i isn't prime, and r contains the remainder of that division.

The loop starts with 3 as the first value to check and continues until i is greater than 100. The variable j starts with 3 as the first divisor to try and the first remainder is calculated. The program then loops until a divisor that evenly divides in i is found, or j becomes greater than the square root of i. If this happens, then i is proved to be a prime number.

If i is prime, that fact is output, and the next odd number is taken in an attempt to prove that it is prime.

C programs can be entered using the standard Assembler/Editor text editor. I recommend saving them with a name like DSK1.PRIME:C. Appending the :C is a way of telling that it is a C program later when you are looking at a directory of the disk. It also will be useful when we are compiling the program.

When we execute a BASIC program, there is a program running in the computer called an interpreter that scans the BASIC program and performs the operations that the statements in the BASIC program request. This is very efficent as far as saving memory, but is slow.

The C compiler, on the other hand, is a program that executes in the computer and reads the statements of a C program and produces an assembly language program that will perform the requested operations. Since the program will be in assembly language, it will execute very quickly as opposed to the BASIC program.

To run the Small C compiler, use option 5 from the assembler editor cartridge, RUN PROGRAM FILE. When it requests a file name, enter "DSKx.C99C", depending on which disk drive the Small C diskette is in. After the compiler is loaded, it will ask "Include c-text?". Respond with N for now. It will also ask "Inline push code?". Respond with N for this too.

It will then ask for an input file. Give it the name of the file that you saved the program in (DSK1.PRIME:C). It will then resquest an output file. I think that it is best to give it a name similiar to the orginal name, except append a :I to it. (DSK1.PRIME:I). Here :I stands for intermediate. This file can be deleted later.

If there are errors in the C program, then the compiler will complain. It will display the line where it believes there to be a problem and then will point to the place where the problem occurred with a caret. Finally it will give an error message describing the problem. You'll have to press <ENTER> to get the compiler to continue. Make a note of the line and the problem so that you can correct the problem later. If there doesn't appear to be anything wrong with the line, then the problem probably occurred on the previous line. I can tell you now that the most frequently occurring problem in C programs is forgetting the ; after statements, and the compiler never notices this until the next line. Simply reedit the program after the compiler completes to fix the problems and then recompile.

When the compiler executes and finds no problems, it will produce an assembly language source program in the output file. This will have to be run thru the /4A's assembler to produce the final executable program. So load the assembler, specify the compiler's output file as the source file (DSK1.PRIME:I), and then specify an object file. Here I recommend appending :O (DSK1.PRIME:O) to distinguish it from the other files. Don't bother producing a listing file and no options are necessary.

If there were errors assembling, reexamine the C program for errors. It may be necessary to look at the assembly language program for clues.

Assuming all went well, you are now ready to execute the program. Use option 3, LOAD AND RUN, and enter the name of the assembler object file (DSK1.PRIME:O) as the first file to load. Next load the C support routines. They are on the C diskette and are called DSKx.CSUP. Finally, for the PRIME program to run, you must load DSKx.PRINTF.

After the last object file is loaded, press <ENTER> at the FILE NAME prompt and you'll be prompted for the program name. Type the letters START but don't press <ENTER> yet!

Take a deep breath, buckle your seat belts and now press <ENTER>.

Zoooooooooooooooooooooooooooooooooooom!

The prime numbers between 3 and 100 will be generated faster than you can see them!

In the next class (article), we'll try to cover less ground with more details. We'll discuss functions, input, and output.

Till then, type in the prime number program above and at least go thru the steps of compiling and executing it.

Mike Schultz

PS: Holy Moley! There's a bug in the example program. If you look real quick while it is running, you'll notice that it doesn't generate 3 as a random number. See if you can fix it.

MS

<u>Central Texas 99/4A Users Group</u>
<u>General Membership Meeting</u>
<u>July 10, 1986</u>

The July meeting of the Central Texas 99/4A Users Group was held on Thursday, July 10, 1986 at the Commissioner's Court Room of the Travis County Courthouse Annex building. The meeting was called to order at 7:55pm by the President, Joe Pizza.

The minutes of the June meeting were read by the Secretary, Mike Schultz.

The Treasurer, Paul Dunn, made a treasury report. There was discussion about the costs of the Austin Computer Faire verses the new memberships.

Mark Milam announced that he got a source for diskettes and that they were available for sell at $7.00 for 10.

The BASIC SIG was discussed.

Joe Pizzi announced that mailing of the newsletter before meetings would resume next month.

The phone number of Mark Milam's bulletin board, The Lightpole's Hideout, was given out. The number is 512-339-1822.

There was discussion about the Assembly Language SIG's classes in C.

There was discussion about Freeware.

Mark Milam, the librarian, distributed a list containing libraries contents.

There was discussion about the charge for getting programs from the library. Mark Milam made the following motion concerning charges. "Programs from the library will be charged at the rate for $1.00 for each cassette and $2.00 for diskette. The librarian will put as many of the requested programs on the requsted media as will fit." William Peale seconded the motion. The motion passed with no decenting votes.

The meeting adjourned at 8:35pm.

Mike Schultz
Secretary

From the Editors Keyboard

I wish to report on the questionaires that I have received to date.

Category	#	total %
Basic/XBasic	7	53%
E/A	9	69%
Hardware	6	46%
TI-Writer	2	15%
Forth	4	30%
Adventures	2	15%
Multiplan	4	30%

Also, the number of persons wanting weekend classes were 1, and 7 wanting weeknight. Three said either would be fine.

Mark Milam
Editor

Who We Are:

We are a non profit organization whose membership is open to anyone interested in the activities of the group. All members pay annual dues of $15 . The membership year is one calendar year from receipt of your dues. You are invited to attend a couple of meetings before deciding if you wish to join.

OUR MEETINGS:

Our meetings are on the second Thursday of each month, at 7:30 PM, in the County Courthouse at the corner of 10th and San Antonio, on the second floor. Each meeting starts with club business and is followed by a demonstration or talk. The second meeting of the month is our special interest group, the Assembly Language SIG. Everyone is urged to share information on topics related to the TI 99/4A -- software review, hardware availability, programming tips, etc.

MEMBERSHIP:

You may join at any time of the year. Each membership "unit" has one vote in club matters and only one person from a "unit" may hold club office (on the other hand your entire family is invited to serve on club committees and participate at meetings and in meeting planning!). Our monthly newsletter is available to members at the first monthly meeting and is sent to those who don't attend. At meetings, you may buy 5 1/4 " disks for $7 each box. We buy these items in bulk quantity and charge cost plus enough to keep several in our library.

OUR LIBRARY:

Currently, our library is located at the librarian's house and will be available at each of our meetings. Our library has some 350 programs on disk and tape. Roughly half of the library's programs will run on just the console (some may require joysticks) and most of the others only require the Extended Basic module. We have programs of all types (games, education, scientific, and business and household management) and we poll the membership's interests before purchasing more. Above all, we are a membership organization. We depend on everyone for directions the group should go and activities we should undertake. Join our group and share your ideas!

MEMBERSHIP STUFF
WELCOME

We wish to welcome to our group the following new members:

Joyce Statz
Spruce R. Keen

And we'd like to thank the following persons for renewing their membership:

James Knox
Bruce Griffiths

Finally we'd like to remind the following people that their membership will be expiring this month and ask (beg) them to renew:

Linda Oakes
Anthony Stephenson

Mike Schultz
Secretary

Commercial Ads

Commercial advertisements are welcomed by our newsletter. This newsletter can provide a select, specialized audience for advertisers. Advertisements also help our group by offsetting the printing and mailing costs of the monthly newsletter. Any advertisment must arrive by the first of the month to be included in that month's newsletter.

The cost of placing a full-page ad is $20. The cost of a half-page ad is $10. The ad should be photocopy ready. Some flexibility is allowed in the size of half- and full-page ads--but let's not overdo it!

Classified ads are free to individuals both members and non-members. Send your ads to Central Texas 99/4A Users Group; Box 200246; Austin, Texas 78720-0246

Classified Ads

WANT TO ADD 32K BUT DON'T HAVE A P-BOX? For a mere $35.00 I'll put 32k directly into your console or your speech-synthesizer. You supply console or speech, and cash and I'll do the rest. Call Mark at 836-3301 after 6:00 PM or drop me a note,

Mark Milam ,4203 Yucatan, Austin, Texas 78727-5967

Equipment for sale!!

Brian K. Neidig Rt.2 Box 157 CC Taylor,Tx 76574	TI 99/4a System for sale-includes console, 2 disk drives, 32k memory, and PEB. All in EXCELENT condition. I also have 20 modules, 40 diskettes, magazines, and several books. This entire system cost me over $2200. I am asking $??? (NEGOTIABLE). I will consider offers on pieces. Call Brian at (512) 352-8132 (Taylor) after 6 p.m.
Zack Swenson 836-2726	Black console, speech, PE Box, 32k, RS-232, 2 disk drives, software, and a printer (needs repair)
Rick Prekap 255-1065	Console, game carts (3 ea) $35 for all!!
Mike Green 1604 Magic Hill Pflugerville,Tx. 251-3684	Complete TI system for $550.00 call or write for more details.
R.L. Tipton PO Box 1198 Rockdale,TX. 512-446-2910	Console,32K memory expansion, Percom disk drive, several games, desk, and tv display, $400 takes it all
Richard Glass 12401 Beartrap Austin, TX. 258-2892	TI LOGO II; $25, Editor/Assembler package;$20
John K. Strickland 12717 old Anderson Mill RD. Austin,TX. 78726 266-2788	Console with cassette cable and player, Asking $50
Craig Deere 8104 Forest Mesa Austin,TX. 78759 346-2609	Console, with TI cassette player. $50, (Extra software available)
Sue Lacey 892-4535	Console, and some software,Flexible $

Ed Eakin (817)699-1368 (answering machine) (817)634-5626 (mon,wed,fri)	Complete TI system, console,PEB,32K,RS-232,Disk controller,Disk Manager, and TI-Writer. First $250 takes it all.
Diane Leonard 451-1702	Two complete P-Code systems for sale! Leave name and phone number if interested
Mike Elder 288-2620	TI-Writer+Spellchecker,$25;Personal Recordkeeping $6 ea.;Astrology chart and biorythm (disk)$10;Dow 4 Gazelle $10;Flip Checkers (disk)$5.

That seems to be all this month, if I have made any errors, please let me know, also let me know if the equipment you have listed is sold, or if you want it run in the newsletter next month.

Mark Milam
Editor

Current Officers

President	Joe Pizzi	444-6829
	1300 S.Pleasant Valley Rd.,#121,Austin, Texas 78741	
Vice President	Al Caldwell	327-8462
Treasurer	Paul Dunn	258-4308
Secretary	Mike Schultz	835-2377
Librarian	Mark Milam	836-3301
Newsletter Editor	Mark Milam	836-3301

Correspondence for the group can be sent to:
the Central Texas 99/4A Users Group
Box 200246; Austin, Texas 78720.
or to the address given for the president.

Meetings Calendar

The following is a list of the currently scheduled meetings.

August 14 September 11 October 9 November 13 December 11

The meeting is scheduled to be held in the County Commissioner's Courtroom on the second floor of the Travis County Courthouse annex, which is at 10th and San Antonio, in Austin Texas.
Meetings start at 7:30 PM and last until they throw us out!

The Assembly Language / Small C / SIG is held the Wednesday following the general meeting,at the Healthcare International building on Great Hills Trail off of 183 just north of 360.

Newsletter Exchange

Our users group exchanges newsletters with several other recognized 99/4A Users Groups. The exchange is made with the understanding that, with proper credit to both the newsletter and author (if listed), your users group can reprint articles from our newsletter and, with proper credit, we can reprint articles from exchanged newsletters. (Please feel free to correct any typos, misspelling, bad grammar, etc.; we will do the same.)

Please send your exchange newsletters to:

Central Texas 99/4A Users Group
P.O. Box 200246
Austin, Texas 78720-0246

Central Texas 99/4A Users Group
P.O. Box 200246
Austin, Texas 78720-0246

AUSTIN TX 787
PM
11 AUG
1986

USA 22

Edmonton 99'er Computer UG
P.O. Box 11983
Edmonton Alberta
Canada T5J 3L1