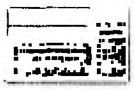
Central Texas

Sept 86

CENTRAL TEXAS 99-4A USERS GROUP THE PAPER PERIPHERAL



From the Kaptains Keybaord

WOW! I can't believe it. One of our members suggested a topic for the next meeting. We were so overwhelmed we decided to do it! So at the next meeting of the group, come and see "The Lightpole's Hideout". A bulletin board program written by The Lightpole and Kaptain Kludge.

We'll have two systems there, one running the bulletin board, the other acting as a terminal. Now those of you who don't know anything about telecommunicating can see what it is about, those of you who have never called the Hideout can see what it is like, and those of you who call it frequently can see what it is like on the Sysop's end.

Be there or be square.

(The President, Joe Pizzi, is out of town this month. His articles will return in October.)

Mike Schultz
The Kaptain of Kludges

Programming in C
An Introduction - Part 2

This is the second in a series of articles concerning the Small C language, which is now available for the /4A. It is hoped that if you are interested in Small C, you'll also come to the Small C classes that the Assembly Language SIG is having. This way you'll get to ask questions. I also recommend that you get a book on C from the bookstore. A book will go into more detail about C in general than I can here.

Last month we discussed most of the basic features of Small C. We still need to cover a few more features before we can back up and cover individual items in detail. But before we do that, last month I left you with a question so lets get that out of the way. As you'll recall, last month an example program was given that generated prime numbers. Unfortunately, it did not find 3 as a prime basically because it assumed that 3 was prime and never tested it. My question was how to fix this. The only response that I got was from a friend at work, Dave Wolfe. His feeling was that since the algorithm used in the program assumed that 3 was prime, it could simply print 3 as the first number and be done with it. Since we would need special code in the loop to detect 3 as the only special case, he may have a point.

Now on to this month's topics: functions and I/O.

The idea behind functions in C is similiar to that of GOSUBs in BASIC. When a function is encountered in a C program, execution of the program passes to the function which executes, performs its job, and then returns to the original program right after the call to the function. However, this is the extent of the similiarity between the two.

One difference between them is that C uses the name of the function to call it instead of using a line number. This is better because the name of the function can be more descriptive of the operation that it performs than a line number. (Besides C doesn't use line numbers!)

Another *difference* is that C functions can return values and thus can be used in expressions. Here's an example:

```
y = 2 * random() + 1;
```

The function random generates a random number and returns it. When C encounters the function while it is evaluating the expression, it temporarily halts the evaluation to call the function. When the function is finished, it returns the value which is then used to complete the evaluation of the expression.

Notice how the function call looks very similiar to a variable name. One might even think it is an array reference, but remember that C uses []'s for its subscripts.

Another nice thing about C functions is that information can be passed into them in the parenthesis. They can use the information, called parameters, to produce the value returned. For example, let's say that a function exists that calculates the square root of a number. The number would be passed into the function as a parameter.

```
y = sqrt(x);
```

Now, don't think that functions may only be used in expressions. It is possible to use functions all by themselves to accomplish a necessary job, that you might not really want to see the details of in the middle of you program. For example, say that we need to generate a tone in the program. Then a function could be written to do this and it would be called everywhere it is needed.

```
beep();
```

Note that the parenthesis are still required by C. That is because even when used by itself, a function can still have parameters. (To vary the lenth of the beep for example.)

Now let's look at an example of how a function is written. Let's say that we need a function that will return the lesser of two values. A call to it might look like:

```
b = min(x, y);
```

The function itself could be written as:

```
min(c, d)
int c, d;
{if (c < d)
     return(c);
    else
     return(d);
}
```

Notice the declaration of c and d as integers. They are the variables within the function that contain the values of x and y which were passed in as parameters.

It is very tempting at this point to go on and talk more about writing functions, but I think that it would be better it we looked at the way that Small C handles I/O. At the same time we could look at some more examples of function calls, since all the I/O in C is handled thru functions. (We'll save writing functions for next month.)

In BASIC, when we opened a file, we told BASIC what file number we wanted to use for the file. Then, when we did any I/O to that file, we used the file number in the INPUT or PRINT statement. A similiar thing happens in C, except instead of us telling C what file number we wish to use, the open of the file picks an available file number and returns the value to us. Here's an example:

```
int unit; /* These are variables to be used */
int i; /* in the following examples */
int k, unit2;
char a[10];

unit = fopen("DSK1.FILE", "r");
```

This call to fopen will open the file as INPUT or read only. The "r" accomplishes this. The second parameter of fopen is used to specify all the open parameters. You can see a complete description in the Small C reference manual that came with the compiler, but some of the other possible open parameters are: "w" for OUTPUT or write only (this deletes and recreates the file), "u" for update (allows both reads and writes), and "a" for append (allows writes starting at the current of file).

If fopen could not open the file for some reason, then it will return a 0 for the unit number. A program should always check for this before preceeding.

If the value returned by fopen is not 0, then the open suceeded and the value returned by fopen, and in this case placed in the variable unit, can be used to perform I/O to the file.

```
i = fread(a, 10, unit);
```

The function fread reads the file specified by the file number contained in the third parameter, in this case unit. The first parameter is the place in memory to place the data read, and the second parameter specifies the maximum number of bytes to read. Now fread will only read the number of bytes that are actually in the next record, and will return this number as its value. In this example, that value is placed in the variable i.

This value can be used to control the action of the program as it processes the data read. Let's say that the program simply wants to write the data back to another file then it could use i in the fwrite to do that.

```
k = fwrite(a, i, unit2);
```

Once again, the first parameter is the location in memory where the data to be written is stored, the second parameter is the amount of data to write, and the third parameter is the unit number of the file on which to write the data.

The function fwrite returns the number of characters actually written, but this is usually the number of characters requested.

It is often useful in a program to be able to detect the end of data in a file, and a function is provided for this also:

```
feof(unit);
```

This function returns TRUE if the last record in the file has been read. The function eof is usually used in a if or while statement.

Finally, when we finish with a file, we must be sure to close it.

```
fclose(unit);
```

This function returns false if the close fails. This usually only happens if the file is already closed and few programs check for it.

Now, let's put all of this together with a simple program that copies an file named INPUT to another file named OUTPUT.

```
#include "DSK1.STDIO"
main()
{char buf[80];
 int len;
 int unitin,
     unitout;
 unitin = fopen("DSK1.INPUT", "r");
 unitout = fopen("DSK1.OUTPUT", "w");
 while (!feof(unitin))
    {len = fread(buf, 80, unitin);
     fwrite(buf, len, unitout);
    };
 fclose(unitin);
 fclose(unitout);
}
```

The only thing I haven't discussed yet is the #include instruction at the beginning of the program. #include is essentially the same as the COPY directive in the assembler, or .IF for TI-Writer. Or, for those who know Extended BASIC, it is similiar to the MERGE command except that the merge is done only during the compilation of the program and never gets put directly into the source.

The include file STDIO is contained on the Small C diskette with the compiler. It contains all the necessary definitions to allow our programs to access the Input and Output functions that Small C supplies.

I have mentioned just a few of the Input and Output functions available in Small C. I suggest that you look at the Small C reference guide that came on the compiler diskette for more information. There is one very useful function that is available on the Small C disk however that is not very well described and that is printf.

The input and output that we have been discussing in this article is essentially unformatted. That is, data is transfered to and from the file essentially as bytes and as is. If we have a value in, say an integer i, then if we try to output directly from i using fwrite, then the internal binary value of i would be transfered, not the ASCII characters that we can read.

The printf routine acts like the PRINT command of BASIC. Taking the internal value of the variable, converting it to essentially display format and then writing it to the file. As is typical with C however, the way this is done is different than it is done in BASIC.

Here is an example of a call to printf.

```
printf("The answer is %d", i);
```

The first parameter to printf is always a COMPLETE description of the output record. It is similiar to the IMAGE statement in Extended BASIC. The rest of the parameters in printf are the values to be formatted and printed. They are placed in the output line in the locations indicated by the %d and other % format codes. %d indicates that the value is to be formatted as a decimal value, %c indicates that the value is to be printed as a single character. %o is for octal format, %s is for string, %u is for unsigned decimal and %x is for hexidecimal.

If a number appears between the % and format code, that specifies the field width. For example %5d states that if the formatted number takes up less than 5 characters on the line, then leading spaces will be added by printf to make it fill 5 characters. This is very nice for tables. If the number between the % and format code starts with a 0, then leading 0's will be added.

A few notes about sprintf. It doesn't check that the parameter that you are asking to convert is really an integer or character, it also assumes that for every format code, there is corresponding a parameter passed for it.

Finally, in order to use it, you must have the following statements in your program:

```
#asm
 REF PRINTF
#endasm
```

And when you are loading the program to execute, be sure to load PRINTF off the compiler disk.

One last thing that I would like to touch on before we're finished. You may have noticed that string is one of the format codes for printf. C does have the concept of strings, but it is different from BASIC (again).

The main idea behind strings in most languages is that a string is a series of characters, the number of characters in the string being of variable length. In BASIC, management of strings is mostly handled by the BASIC intreperter. In C it is left up to the programmer. To have a string in C, one merely declares a character array, place the individual characters of the string into individual array elements, and place a binary zero in the element after the last character. This means that the programmer is responsible for ensuring that the string being placed in the array is not too big.

When we use a literal string in a C program such as "The answer is %d", the C compiler will set aside space for the characters in the string, place the characters in it and also add the binary zero.

I mention strings here because it is one of the things that C handles best (I know that that is hard to believe).

You may be wondering why I did not mention interactive I/O here. Well that is because I'm having a little problem getting it to work. But I'm sure that I'll have more time next month and will pass the solution along to you then.

C you later,
Mike Schultz

Central Texas 99/4A Users Group
General Membership Meeting
August 14, 1986

The August meeting of the Central Texas 99/4A Users Group was held on Thursday, August 14, 1986 at the Commissioner's Court Room of the Travis County Courthouse Annex building. The meeting was called to order at 7:35pm by the President, Joe Pizza.

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

The Treasurer, Paul Dunn, was unable to make a treasury report, because the Secretary did not bring the necessary statements.

There was discussion about the BASIC SIG. Joe Pizzi stated that the problem with getting the SIG started is that most people wanted it on the weekend and finding a meeting place on the weekend is difficult. Mark Milam stated that of those people who returned the newsletter survey forms, the ones who wanted a BASIC SIG stated a week night was perferable.

Mike Schultz asked the membership if the C article he wrote for the last newsletter reached any level of understandability.

There was discussion about the AL SIG.

Mike Schultz suggested that the group organize a party.

Joe Pizzi read letters sent to group from Harry Allston and Mark Beck. Harry's letter was a request for help from Users Group members to train handicap people to use computers, specifically the /4A. Mark's letter was to announce the availablity of the latest release of CSF, a database program. CSF is a Fairware program and proceeds from the sale of it go to the Jacksonville Users Group to support their newsletter.

Mark Milam talked about the programming contest that has announced in the Computer Shopper magazine.

There was a discussion about the Aminion Helpline.

The meeting adjourned at 8:02pm.

Mike Schultz
Secretary

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:

Rodger Quintanilla

Kieth Johnson

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

(none this month!)

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

Jerry Tindle
Jin-Gen Yang
John Young

```
---------------------------------------
!                                     !
!            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 936-2726	Black console, speech, PE Box, 32k, RS-232, 2 disk drives, software, and a printer (needs repair)
Cal Emery 209 Live Oak Harker Heights,Tx. 76543 (817)699-5382	Black console,PE Box with TI RS232 card, speech, joysticks, recorder cable, various manuals and cartidges. Everything in excellent condition. Total price:$200.00
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	327-6462
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.

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
PM
SEP 6

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