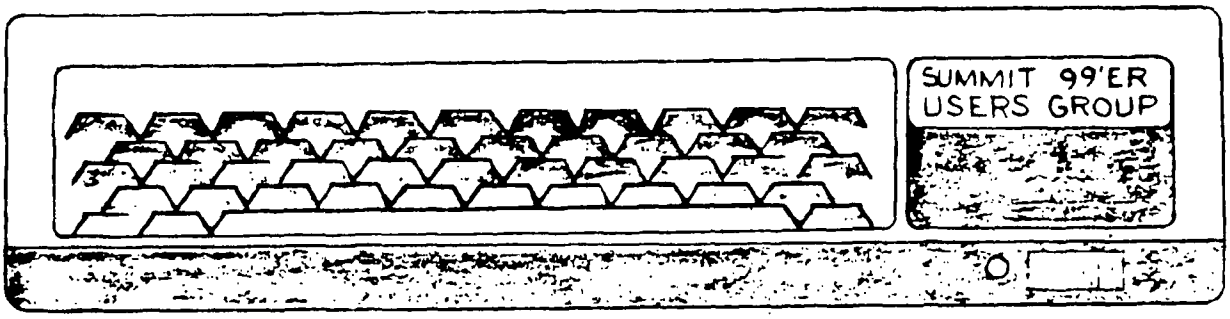

JUNE 1984 Vol. 2 No. 6

The June meeting will be held on Thursday, **June 21** at Cuyahoga Falls High School on Fourth and Stow St. in Room 413 - Physic's Lab. The July meeting will be held on **July 19**. Please remember to sign in.

PLEASE BRING YOUR LIBRARY TAPES IN.

This month's program will be a **SWAP MEET**. You will be able to check out library tapes for that night's use from 8:00 to 9:30. We are asking you to bring your keyboard, monitor, cassette player, dual cables, power strip and/or extra extension cords if you want to make copies from our library. If you can't bring in your setup, at least bring in your dual cables, cassette player, power strip and/or extra extension cords. You can only check out one library tape at a time, then when you return that one, you can take out another one. If you need blank tapes we will offer C-10 for .75 each (this is a special one-time price). If you want to check out tapes for the month of July you can do so after 9:30. If we have a good turn out this month, we may run the SWAP MEET again in July or August.

POSITIONS AVAILABLE

Paul Hayden, who was our first President, now serving as Secretary, has stepped down from his position. We thank him for his service and hope he will still remain an active member.

Tom Sanders, our Program Vice-President, has had to resign from his position due to conflicting work and family schedules. We will miss his great programs and hope that he will also remain as active as possible in our group.

Due to these circumstances, we are looking for a Secretary and Program Chairman. If you would like to take an active part in making our group grow, why not volunteer for the position of Secretary or Program Chairman. For details about the responsibilities of either of these positions, call Pat Bowen at 920-1884 or me, Kathi Anderson at 923-7530.

WE NEED ARTICLES

If you want to share a programming tip, review a piece of software or hardware, or have written programs you want to share with us, we would like to know what you are interested in. We need input from you.

This article comes from the May 1984 issue of Spirit of 99, out of Columbus, Ohio.

COMPANION VS. TI-WRITER

By Niraj N. Shah

This is a review /comparison of two word processors named COMPANION and TI-WRITER. COMPANION was developed by Intelpro and TI-WRITER by Texas Instruments. The thing that makes the COMPANION unique from most word processors is that a lot of it is written in Assembly Language. The only part that is written in Basic is the system monitor for COMPANION. TI-WRITER is written entirely in Assembly Language. COMPANION is available only in a Disk form and TI-WRITER is available in a module and disk form. The COMPANION is available for $79.95 (U.S.A.) at this address:

INTELPRO
5825 BAILLARGEON STREET
BROSSARD, QUEBEC, CANADA
J4Z 1T1

The COMPANION is the only word processor on the market can be compared to TI-WRITER (that I know of). Keep in mind that each has its advantages and disadvantages. I will try to point out the differences between the two word processors in an objective manner.

Obviously, the main difference and the most important is that COMPANION has an 40 column editor where TI-WRITER has a variable column (maximum of 80 columns) editor. If you need to see how the text is going to look on the paper prior to printing then you should choose TI-WRITER as your word processor. The second major difference between the two is that COMPANION has to be loaded into memory only once, no swapping of disks! TI-WRITER forces you to swap disks (for one disk drive systems) if you desire to use Text Formatter. But remember because of this second major difference TI-WRITER is capable of handling more text than is the COMPANION. The third major difference between the two is how they print out the text onto a printer and what options they provide. COMPANION does NOT provide a right justify command where the TI-WRITER has that command. Intelpro has stated in their documentation that right justification is in being worked on in their R/D department.

Both processors have the capability to move, copy, and delete blocks of text. COMPANION deals with blocks of text via special indicators. TI-WRITER does it with line numbers. The documentation for both processors is quite extensive and easy to understand. Thus, no matter which word processor you choose, you most likely will not be left in the dark. But as usual, experimentation is the only way to find out what you can and cannot do with the word processor. Both word processors can do global or selective search,replacement and deletion of strings.

The COMPANION has two added significant features which TI-WRITER does not have. COMPANION can count the number of occurences of a unique string within a body of text. The second feature is that COMPANION lets you customize all the defaults for printing via a program on the program disk. But keep in mind that the cusomization invalidates the warranty! Both word processors have unique symbols to indicate the beginning of a paragraph, line feed, center a string and start a new page. But the COMPANION also has a one keystroke command that enables one to tab over a specified number of columns for things such as salutations in a letter.

The thing that COMPANION is sorely missing is an OOPS! feature, which recovers inadverdantly deleted text. It also needs a Delete-To-End-Of-Line keystroke command. I would also like to see Intelpro add a Delete Line command in future revisions of the program. COMPANION also needs some way to move the cursor in a vertical motion through the middle of the text. In other words, when I use the Up Arrow key in COMPANION it will eventually left justify over to the first column thus making it almost impossible to see if the columns of two tables are even. TI-WRITER

does not have this problem; when changing rows the columns do not change and vice versa. The labeling of the function keys in COMPANION seems to be rather backward. To scroll down 12 lines I must press <FCTN 4> (CLEAR) and to scroll up I must press <FCTN 6> (PROCEED). To me, the word PROCEED means to go to the finish or the end of the text. But not so in Intelpro's interpretation! Also, to use the Editing commands in COMPANION I must press <FCTN 9> (BACK) and to get the the main menu I must press <FCTN 7> (AID). To me the word BACK means to go back to the main menu. Again, Intelpro seems to have their English mixed up.

Never fear, COMPANION does have some advantages over TI-WRITER. The main advantage is the considerable speed in which COMPANION deletes, moves, and copies blocks of text. The speed difference is quite obvious when using the auto-repeat feature of any key being pressed is used. If you hold down the left arrow key then you will see the cursor move to the left at normal speed then accelerate to the speed of light! This is quite evident when deleting characters, the letters move so fast that they become a blur! The COMPANION has three different symbols for the cursor: Edit, Text and Insert. When inserting characters into the text the COMPANION does not split the line into two sections, before and prior the cursor. The COMPANION's insert feature is a normal one, just like the TI BASIC editor. The lower case letters in COMPANION are true lower case, not small capitals as they are in TI-WRITER. The scroll up and down blocks of text in the COMPANION consists of scrolling 12 lines of text in a smooth but fast manner. TI-WRITER scrolls in an instantaneous scroll of 24 lines of text. When typing text into the COMPANION the text is never re-configured by the program unless you specifically tell it to do so. So, when you reach the end of the line in the middle of a word, COMPANION will merely wrap around to the next line. The word is NOT shifted to the next line as it is in TI-WRITER. This was done to fully utilize all 40 columns of the screen.

Both word processors enable you to send all the ASCII characters (0-255) to the printer. Both allow you to change the background color of the screen. But unfortunately, COMPANION adopted white for the color of the foreground of the text. So, if you have a Black & White T.V. then you may have a problem in reading the text on the screen. I know that I certainly do! Both programs enable you to print text in a single sheet or in a continuous form fashion. Both processors have a warning message to signal impending overflow of the text buffer. But COMPANION also tells you how much text is currently in the buffer, thus, you can keep an eye on how much is in the buffer.

Personally, I think that the COMPANION has a better printer formatter and the TI-WRITER has a better text editor. Unfortunately, the two cannot be merged to form the ultimate word processor. Due to TI's decision to drop out of the market it may be hard to acquire a TI-WRITER. Intelpro is going to continue marketing the COMPANION. Also, Intelpro is willing to personalize their product just for you and based on that fact I think it would be wiser to buy the COMPANION. Keep in mind the minimal sytem configuration for either word processor is Extended Basic, Disk Drive, 32K memory and an optional RS232 and printer for the COMPANION. Replace the Extended Basic with the TI-WRITER module for the TI-WRITER word processor.

This article comes from the HUGGERS HOOSIER USERS GROUP, MAY 1984.

BEST OF THE NEWSLETTER

THE HEART AND SOUL OF PERSONAL RECORD KEEPING

by Don Donlan

Those HUGgers who subscribe to the 99'er HCMagazine probably welcomed the 'CALL PEEKV and CALL POKEV' information related to the Mini-Memory Command module. With this article I would like to begin sharing some of the "secret" subprograms that are a part of the PERSONAL RECORD KEEPING module. A subprogram is an independent program which usually performs only one task or function. Because the subprogram performs only a single or rather narrow task, several subprograms are usually bundled together when making a program or, as in this case, a command module. Subprograms allow a programmer to perform tasks over and over again using the same code. A subprogram is like a meat grinder--it performs one task. But depending on what you 'feed' the meat grinder, you get different results. If I put pork in my 'meat grinder', I would get sausage; with beef, I get hamburger; with ham, I get the base for ham salad. Similarly, you can 'feed' a subprogram differing information to get different results. The information you 'feed' a subprogram is contained in PARAMETERS. These parameters are like serving trays that carry information back and forth between the main body of a program and the subprogram you select to use. Since you probably have more than one subprogram (or more than one task or function you wish to accomplish), you need to be able to select a particular subprogram. A name is given to each subprogram. In this way the computer knows which task or function to perform when you CALL the subprogram by name. Pages 71 through 90 of the TI 99/4A USER'S REFERENCE GUIDE that came with your computer tell about the color graphics and sound subprograms that are part of TI BASIC. You're probably already using the CALL CLEAR subprogram when you want to 'blank out' or erase the screen. Perhaps you've used the CALL SCREEN subprogram and passed along the required parameter (that is, a number from 1 to 16). The CALL SOUND subprogram has the ability to receive nine (9) parameters, but only three are required for the subprogram to do its job of producing a single tone. These subprograms are explained with examples in the USER'S REFERENCE GUIDE. They should give you some basic understanding of how subprograms work. You may want to start there.

For now let's return to the task at hand: defining and describing the subprograms that are available in the PERSONAL RECORD KEEPING cartridge. There are seven subprograms in the PRK command module. A program written in TI BASIC can use these subprograms if you plug in the PRK module and take option 1 (TI BASIC) BEFORE you try to load any program. I will go into more detail in following articles, but at this time I merely wish to list the seven subprograms, their parameters, and some comments on the function they can perform. In the final article of this series, I will be including a TI BASIC program that can be used on a system with a disk drive to access the records created by the PERSONAL RECORD KEEPING command module. This will provide greater flexibility in the use of this cartridge.

SUBPROGRAM NAME	PARAMETERS		DESCRIPTION OF FUNCTION
PREP SUBPROGRAM Code as: CALL P(V)	V	The number of bytes of characters you wish to reserve for use.	Prepares a work area or space that is to be used for the storing of information. Allocates a data area in VDP RAM.
LOAD SUBPROGRAM Code as: CALL L(V$,V)	V$ V	Data file name. Return code.	Loads a data file from a disk and indicates , in the return code, whether the subprogram successfully loaded the file.
SAVE SUBPROGRAM Code as: CALL S(V$,V)	V$ V	Data file name. Return code.	Saves a data file from the work area prepared by the PREP subprogram to a disk. Return code tells whether the SAVE subprogram did its job OK.

THE HEART AND SOUL OF PERSONAL RECORD KEEPING, cont'd

SUBPROGRAM NAME	PARAMETERS	DESCRIPTION OF FUNCTION
ACCEPT SUBPROGRAM Code as: CALL A(Y,X,W,C,V,L,H) or CALL A(Y,X,W,C,V) or CALL A(Y,X,W,C,V,F) or CALL A(Y,X,W,C,V$)	Y Row number. X Column number W Field size or the number of characters C Return code. V Numeric variable. V$ Character variable. L Low value in a range of numbers. H High value in a range of numbers. F Field number of a piece of information within a PRK record.	Works much like the Extended BASIC 'ACCEPT AT' code. It 'captures' information from the keyboard as you key it and then echoes or displays that information on the screen for you to see. Various parameters may be used, depending on whether you want to 'read' numbers, numbers within a certain range, or a character string. The return code tells whether a valid, invalid, or function key was pressed. The F or field parameter can be used to validate data entered ONLY if your program has a second type of record--a header record--which it uses to cross-check the information you enter.
DISPLAY SUBPROGRAM Code as: CALL D(Y,X,W,V) or CALL D(Y,X,W,V$) or CALL D(Y1,X1,W1,V1, Y2,X2,W2,V2$, Y3,X3,W3,V3,...	Y Row number. X Column number. W Number of characters or field width. V Number to display. V$ Characters you want to display.	Similar to Extended Basic's 'DISPLAY AT' operation, this subprogram places either numbers or characters on the screen. Row number ranges from 1 to 24, while the column number ranges from 1 to 28. Multiple displays can be done or performed with one CALL D.
GETPUT SUBPROGRAM Code as: CALL G(R/W,REC,FLD,V) or CALL G(R/W,REC,FLD,V$) or CALL G(R/W,REC,FLD,MIS,V2) or CALL G(R/W,REC,FLD,MIS,V2$)	R/W Read or Write code. REC Record number. FLD Field number. V Number written. V$ Characters written. V2 Number written. V2$ Characters read. MIS 'Missing' return code.	Retrieves and places information from or into the work area defined by PREP subprogram. The subprogram identifies missing data (information not found) when reading a record. This allows the subprogram to tell the difference between a field that has never been entered ('missing') and a field that is all blanks or all zeroes. Saves space in memory by using this code. Values for these codes will be discussed in later articles.
HEADER SUBPROGRAM Code as: CALL H(R/W,INFO,FLD,V) or CALL H(R/W,INFO,FLD,V$)	R/W Read or Write code. INFO Header record item. FLD Field number. V Numeric variable. V$ Character variable.	This is a core subprogram in PRK module. It is used to establish and maintain a kind of 'data dictionary'. This dictionary defines the kinds of information and the location of the information within the PRK records. Fourteen (14) kinds of information or header record items are stored here. This one record is the key to the entire 'data base' created by the PRK module. It gives the characteristics of each field within a record (name, type, size, decimals (if any)), storage space and position). The PRK file name, date, number of fields per record, the total number of records, the length of the header record itself, and the length of each data record are all a part of this one record.

Because of its central role in organizing the Personal Record Keeping command module, this is where we will begin our article in next month's newsletter.

This article comes from Tenderfoot Basic from the Spirit of 99. April 1984 newsletter.

TENDERFOOT BASIC

BY NIRAJ N. SHAH

I would like to give credit to Curtis Garcia of the South Bay TI Users Group for giving me the incentive to write this month's article on Relational Operators. I apologize to those of you who were expecting a TENDERFOOT BASIC column in last month's newsletter. I was too busy with my homework for OSU to find the time to write my column. For those of you who do not know me, I am an Electrical Engineering student at the Ohio State University. I expect to graduate in June of 1984 and become another member of the unemployed club.

This month I am going to discuss Relational Operators. What are Relational Operators? They are the greater than(>), less than(<) and the equal(=) operators. The term relational implies that the operators show the relation between two operands.

To illustrate what I am saying lets look at an example. Suppose Tom and Dick are bidding at an auction. We want to find out whose bid was the highest. Here is one possible solution to the problem.

```
100 INPUT "TOM'S BID = ":TOM
110 INPUT "DICK'S BID = ":DI
CK
120 IF TOM>DICK THEN 130 ELS
E 140
130 PRINT "TOM'S BID WAS GRE
ATER THAN DICK'S BID"
140 IF DICK>TOM THEN 150 ELS
E 160
150 PRINT "DICK'S BID WAS GR
EATER THAN TOM'S BID"
160 IF DICK=TOM THEN 170 ELS
E 180
170 PRINT "BOTH BIDS ARE EQU
AL"
180 END
```

Lines 100 and 110 ask the user what are the respective bids for Tom and Dick. Line 120 compares Tom's bid with Dick's bid, in particular, it checks to see if Tom's bid is larger than Dick's bid. If so, then the corresponding message is printed by Line 130 otherwise execution skips to Line 140.

Line 140 checks to see if Dick's bid is greater than Tom's bid. If so, then the appropriate message is printed by Line 150. Otherwise, the computer skips to line 160, which compares the two bids to see if they are equal. Again, if they are equal then a message is printed by Line 170. Then the program stops execution by Line 180.

I know that some of you more advanced programmers are wondering why I put Line 160 in there. The main reason for the inclusion of that line is clarity. It is easier for the novice to understand the flow of execution if Line 160 is used. However, if you want to change that then both Lines 140 and 160 have to be changed:

```
140 IF DICK>TOM THEN 150 ELS
E 170

160 GOTO 180
```

Try it out and you will see that both versions accomplish the same thing. But, the first version is easier to understand than the second version.

Now that you have seen a concrete example on how relational operators are used in a program lets examine how the computer evaluates expressions that contain those

CONTINUED

relational operators. When the computer encounters an expression such as, A<B, it evaluates it to see if A is indeed less than B. If that is the case then the computer will replace that expression by a -1. Otherwise if it turns out that A is not less than B then the computer replaces that expression by a zero(0). Here is an example.

```
100 A=5
110 B=10
120 PRINT "A<B IS";A<B
130 PRINT "A>B IS";A>B
140 PRINT "A=B IS";A=B
150 PRINT "A<=B IS";A<=B
160 PRINT "A>=B IS";A>=B
170 END
```

Type in this program and observe the results. In Lines 100 and 110 the variables, A and B, are assigned their respective values of five and ten. Then in Lines 120-160 the computer is asked to compare the two variables and print the results of the comparisons. Since A is indeed less than B but not equal to B only Lines 120 and 150 should result in printing a -1. The other Lines should print a zero(0).

Why is this? As I stated above, if the relational expression is evaluated to be true then the computer replaces that expression with a value of -1 otherwise it is zero(0). Since only the expressions in Lines 120 and 150 are true then only those lines will print a true indicator value of -1. The rest of the lines will indicate a false evaluation of their expressions by printing a value of zero(0).

Now, how can one use this facility in his programs? Well, lets try to minimize the following program. This program scans the keyboard for a key to be pressed. When a key has been pressed then it checks to see if the key that was pressed was a numeric key, the numbers zero through nine. If so, then the program prints a one(1) to indicate a valid key was pressed. But if a valid key was not pressed then a value of zero(0) should be printed.

```
100 CALL KEY(0,K,STATUS)
110 IF STATUS=0 THEN 100
120 IF K<ASC("0") THEN 140
130 IF K>ASC("9") THEN 140 EL
SE 160
140 PRINT 0
150 GOTO 170
160 PRINT 1
170 END
```

Here is a shorter program that makes full use of the facilities of relational operators. But it is also a lot less readable.

```
100 CALL KEY(0,K,STATUS)
110 IF STATUS=0 THEN 100
120 IF (K<ASC("0"))+(K>ASC("
9")) THEN 130 ELSE 150
130 PRINT 0
140 GOTO 160
150 PRINT 1
160 END
```

The key point to be made by the second version is in Line 120, which replaced lines 120 and 130 in the first version of the solution. Lets say that the key pressed was less than ASC("0"), maybe the <!> key was pressed. That means that the expression K<ASC("0") is going to be evaluated to be true and replaced by a -1. So now Line 120 looks like this: (-1)+(K>ASC("9")). Since the <!> key was pressed, the expression, K>ASC("9") is false and thus will be replaced with a zero(0). Which makes Line 120 look

CONTINUED

looks like this: (-1)+(K>ASC("9")). Since the <!> key was pressed, the expression, K>ASC("9") is false and thus will be replaced with a zero(0). Which makes Line 120 look like this: (-1)+(0). But since there is an addition symbol still left in the expression the computer has to add the two numbers. So, now Line 120 looks like this: -1. Since the expression in the IF-THEN-ELSE statement has been reduced down to a -1 value(true) the computer recognizes the IF-THEN-ELSE statement to be a TRUE one. Which causes execution to continue on to Line 130 and print a value of zero(0). Which indicates that an invalid key was pressed.

Here is a summary of the sequence of evaluations that the computer made:

```
(1)  (K<ASC("0")+(K>ASC("9")
(2)  (    -1    )+(K>ASC("9")
(3)  (    -1    )+(    0    )
(4)              -1
(5)  Aha! It is a True Statement!
(6)  So, print a value of zero(0)!
```

Now, suppose that the key pressed was indeed a numeric key, such as <4>. Here is the sequence that computer would go through in evaluating the expression in Line 120.

```
(1)  (K<ASC("0")+(K>ASC("9")
(2)  (    0     )+(K>ASC("9")
(3)  (    0     )+(    0    )
(4)              0
(5)  Aha!  It is a False Statement!
(6)  So, print a value of one(1)!
```

Well, you say that is all fair and good but it only saved one line! True, but lets say that the problem statement was changed to print a 1 when the key being pressed was in the range:

```
0 <= KEY <= 9  OR  A <= KEY <= Z
```

In other words, if the key pressed was between 0-9 or between A-Z then a print a one(1). In the first example we would have to add two more lines. Here is the solution:

```
100 CALL KEY(0,K,STATUS)
110 IF STATUS=0 THEN 100
120 IF K>=ASC("0") THEN 130 E
LSE 140
130 IF K<=ASC("9") THEN 160
132 IF K>=ASC("A") THEN 134 E
LSE 140
134 IF K<=ASC("Z") THEN 160 E
LSE 140
140 PRINT 0
150 GOTO 170
160 PRINT 1
170 END
```

Notice that Lines 132 and 134 were added to the above program. Also, take a close look at the logic involved in the IF-THEN-ELSE statements. The relational expressions are quite different from the previous examples. Here is the second solution which uses the relational operators more efficiently.

```
100 CALL KEY(0,K,STATUS)
110 IF STATUS=0 THEN 100
120 IF (K<ASC("0"))+(K>ASC("
9"))*(K<ASC("A "))*(-1)+(K>A
SC("Z")) THEN 130 ELSE 150
130 PRINT 0
140 GOTO 160
150 PRINT 1
160 END
```

Notice that in this version only Line 120 had to be changed. Line 120 has three major checkpoints:

CONTINUED

```
a)  (K<ASC("0"))
b)  (K>ASC("9"))*(K<ASC("A"))*(-1)
c)  (K>ASC("Z"))
```

Part (a) checks to see if the key pressed was less than zero, which is invalid. Part (c) checks to see if the key pressed was greater than <Z>, which is also invalid. The other invalid range is for any key that lies between nine(9) and <A>, not inclusive. This range is checked by part (b). Lets assume that a key was pressed in the range being checked by (b). Assume the key pressed was <:>. Here is how the computer would evaluate parts (a),(b) and (c).

```
For Part (a):
1)  (K<ASC("0"))
2)       0
3)  Aha!  It is a false value!

For Part (b):
1)  (K>ASC("9"))*(K<ASC("A"))*(-1)
2)  (    -1    )*(K<ASC("A"))*(-1)
3)  (    -1    )*(    -1    )*(-1)
4)  (           1           )*(-1)
5)  (           -1                )
6)  Aha!  It is a true value!
7)  Thus it is an invalid key!

For Part (c):
1)  (K>ASC("Z"))
2)       0
3)  Aha!  It is a false value!

For Line 120
1)  (a)+(b)+(c)
2)  (0)+(-1)+(0)
3)  ( -1 )+(0)
4)       -1
5)  Aha!  It is a true value!
6)  Thus it is an invalid  key!
```

The main point of this way of doing the invalid key checking is to illustrate how one can implement OR and AND functions. Part (a) and (c) were strictly relational expressions. Part (b) was a mixture; both relational and an AND function. It was checking for an invalid key by making sure that the key was greater than <9> AND less than <A>. Then finally Line 120 in itself was an OR function. It was checking to see if the key pressed was less than <0> OR greater than <Z> OR between <9> AND <A>. Thus, OR functions can be implemented with the plus(+) operator and AND functions with the multiply(*) operator.

Go through the same procedure as I showed above in evaluating the expression in Line 120 for a valid key. This time the second version of the program saved me four lines of programming! Thus, if there are a lot of IF-THEN-ELSE statements in one part of your program try to get rid of them by using your relational operators more efficiently as demonstrated above.

Finally, this last program is a subroutine that enables one to move in the arrow directions and also in diagonal directions. The diagonal moves are done by using the <W,R,Z,C> Keys. Otherwise, use the arrow keys without the <FCTN> key! Notice how many IF-THEN-ELSE statements I eliminated by using the versatility of Relational Operators! The routine basically takes place of a Joystick routine if you do not have Joysticks. Just insert this in a suitable place in a Joystick based game and Voila! You have a keyboard based game!

```
100 REM KEYBOARD ROUTINE
110 REM BY NIRAJ N. SHAH
120 CALL CLEAR
130 R=12
140 C=12
150 CALL HCHAR(R,C,30)
160 CALL KEY(0,K,STATUS)
170 IF STATUS=0 THEN 160
180 R=R+((K=87)+(K=69)+(K=82
))+(((K=90)+(K=88)+(K=67))*-
1)
190 C=C+((K=87)+(K=90)+(K=83
))+(((K=68)+(K=82)+(K=67))*-
1)
200 CALL HCHAR(R,C,30)
210 GOTO 160
```

This article comes from the Spirit of 99, April,1984 newsletter.

MAGAZINES IN REVIEW

By Jake Hinkle

Computue-published by Compute Publications Inc. at $2.95. This magazine (Mar. 84) has a variety of computers. they are Adam, Apple Atari, Color computer (Radio Shack), Commador 64 & Vic 20, IBM PC and Jr, Pet/CBm, Texas Instruments, and Timex/Sinclair. For a magazine that started having the TI in it about a year ago. The ranking of articles are as such, Commandore 64-16 articles or programs , Atari-13, Vic 20-10, and in 4th place with 5 is TI. There are 12 other articles/features that are for all 11 machines above. The TI has a regular column by Regena. To my amazement there are only (2) other machines which have there own column. Compute generally has at least one program in basic and one in X-basic. In the Feb. issue they even had a Terminal Emulator II articles on Spanish. This month programs are Roader (B), Aquarium (X-basic) , File processing (B), and sound shaper (B) are the programs. Hopefully in the Library (CONNI) soon. Some of the articles are meant for people with some experience in programming. But for the beginner that wants to learn about programming or to obtain programs cheap, at $2.95 for 4 programs is a good buy at Krogers or Gold Circle and several of the book stores have it also. If you can not find Compute then you might try having the store order it from Scott Krauss News for you.

The main feature of the magazine that intriqued me was the "Guide to Articles & Programs". This tidbit list to the right of the index page what machine each articles is for. And an Astrisk is for all machines (that is the 11 I mentioned earlier). The magazine is divided up and they are: Features-basicly about all machines but sometimes there are articles about a separate machine but Compute feels that it is relevant to all machines. Education and Recreation- are generally about different games. This month "Roader" has 8 of the 11 machines with side by side programs of Roader. Reviews- are of different functions of different machines or products for specific computers. Columns and Departments include "Editors Notes", "Reader feedback", "Programming the TI", ect. etc... The Journal" includes articles on various computers. The last section of of the magazine tells you "How to ENTER Compute's Programs" a must if you are just learning to:type on TI's keyboard. keyboard. "A Beginners Guide to Typing in Programs" which is self explan- atory. "Compute! Modifications or cor- rections to Previous Articles" or as Games magazine puts it "Dirty Laundry". "Product Mart" a small classified. The last section is "Ad--vertisers Index". The mysterious thing about magazines is that for the over priced machines they jump on the bandwagon and pop out a magazine for specific machines. Compute puts out Compute! Gazette for Vic 20 & Com madore 64 and now that IBM introduced the Jr. Compute now has Compute! PC & PC Jr. I cannot tell you how many magazine there are for the PC But I feel that if enough people call or write Compute. that they might consider a magazine dedicated to the 2+ million machines in use. Toll free # is 1-800-334-0868 +or subscriptions. Next month I will cover "Home Computer Magazine" formally 99'er Home Computer Magazine Till next month "Happy Computing". JAKE HINKLE 668-0032

There are 2 wholesale groups in this area we would like to talk about.

1) The Warehouse Club on Arlington Road. Take 77 South to Arlington Road exit, turn right and it is next to Gold Circle. Membership is free if you belong to a credit union or are a retired state or federal employee. If you are neither, I think the membership is $30.00 a year. They charge the wholesale price plus 5 percent. They have great buys on furniture, paper and disks as well as other merchandise. The 3M single sided, single density was $17.50 a box.

2) The Wholesale Club on Rockside Road. Take 271 North to Rockside Road exit, head west for about six blocks. The membership requirements are the same as the Wholesale Club

If you have any questions about either of these groups, call Pat Bowen for more details.

LIST OF BOARD MEMBERS AND THEIR HOME **PHONE NUMBERS**

President, Pat Bowen	920-1884
Vice President, Norm Sorkin	678-2360
Librarian, Leroy Martin	666-3984
V.P. Program,	
Secretary,	
Treasurer, Betty Duncan	633-5217
Educational Director, John Curry	929-8824
Editor, Kathi Anderson	923-7530

If you need another computer catalog in your house, you can write to: Quill Corporation, 100 S. Schelter Road, P.O. Box 4700, Lincolnshire, IL 60198-4700. Phone number is 1 (312) 634-4850.

If anyone would like to write to Michael Noble at: 125 Crestline Drive #701, Clarksdale, Mis. 38614. Phone: 601-624-8467. Drop him a line to say hello.

I would like to thank all the Users Groups I borrowed articles from. See you at the meeting. Kathi Anderson, Editor

SUMMIT'99ers USERS GROUP
Kathi Anderson, Editor
3240 Bailey Road
Cuyahoga Falls, Ohio 44221