The Edmonton 99'er Computer Users' Society
p.o. box 11983
Edmonton, Alberta
CANADA T5J 3L1

99'er ON LINE... is the news letter of the Edmonton 99'er Computer User's Society published ten times a year. Unless otherwise stated, all articles may be republished in other news letters provided that source and author are identified. We will credit authors quoted in 99'er ON LINE.

CORRESPONDENCE: Newsletter editor: Bob Pass, 59 Labelle Cr, St. Albert, Alberta, Canada T8N-2G6. (403)458-7658. All other correspondence should be sent to the address at left.

OFFICERS: president Tom Hall, vice-pres Ken Godbeer, treasurer Jim Mulligan, secretary Roxanne Appelt

REGULAR MEETINGS... of the Edmonton 99'er Computer User's Society are held on the second Tuesday of each month in room 849 of the General Services building of the University of Alberta from 7:00 till 10:00 PM and are open to all members in good standing. Non-members may attend their first meeting free of charge.

ADVERTIZING... Commercial space is available in this news letter at the following rates: full page $20.00, half page $15.00, 1/4 page $10.00. Discuss your needs with Jim Mulligan at 467-6021, at the next meeting, or send "photo ready" copy to the P/O Box above. Members may advertise their personal computer related items for free but are asked to limit their ads to about 50 words. Mail your ads to the editor's address or hand it to him at the general meeting; newsletter deadline 3'rd Monday of the month.

MEMBERSHIP FEES: Family 12 months, $20.00, 6 months, $15.00. Students 12 months, $15.00, 6 months, $10.00. New member initiation, $20.00.

FEBRUARY MEETING

CLUBLINE/99 -- The December and January issues were distributed at the meeting. About 5 extra copies are available at each meeting for general sales at $2.00 each on a first come, first serve basis. Also, back issues of the monthly diskettes are available; place your order with Bob Pass. A show of hands indicated that we will continue with the subscription.

ATTENDANCE -- at the meeting once again exceded 30 members. It is good to see a good turn out for the meetings.

DOOR PRIZES -- We finally got our act together and made our long promised door prize draw. Prizes were a black jack and poker module donated by Tom hall and two packages of vinyl 3 ring binder disk pockets donated by Bob Pass.
GROWLIES -- Ken Godbeer was not able to get to this meeting, so we missed our donuts. But Lois Meunier once again provided free fruit cake so not all went hungry! Thanks Lois.

IBM COMPATIBILITY -- see article elsewhere in this issue.

FREE TO A GOOD HOME -- Yves Chevalier has rescued several rolls of teletype paper from a trip to the dump and will share them with members who can pick them up. You must have a friction feed printer to use these rolls. Call Yves at 456-6887 if interested.

DEMONSTRATION -- Paul Helwig presented an introductory talk on c99 covering the general points required to get the system loaded, how to compile your source file and how to assemble and run the compilation. Paul also demonstrated three programs in c99 and handed out photo copies of the programs so we could try it out our selves. Jim Beck then demonstrated his newest game program which is written in c99 and realy gave us a good view of the speed of the system and how much code can be packed into a single program. Thanks Paul and Jim for an informative presentation.

NEXT MEETING

The next meeting will be Tuesday, March 10'th at 7:15 PM. Same place as usual; General Services Building, U of A campus in room 849.

As of this writing, an agenda for the next meeting has not been set. So it will be pot luck.

NEWSLETTER DISKETTE AWARDS

This month there are two people receiving a diskette honorarium: Jim Beck for contributing three programs in basic and extended basic and to Michal Jaegermann for a second tutorial in c99. Thanks Jim and Michal for your continued support.

As you can see, Jim and Michal's contributions form the bulk of this newsletter. Input is needed from any of you; I am concerned that this newsletter is not addressing the needs of many of you. By writing some articals or sending me some newspaper and magazine clippings, I can make this newsletter more appealing to a broader segment of our membership.

Don't forget too that by contributing to this newsletter you become eligible for a free diskette. So let's see some more input from you folks!

FIRST STEPS INTO c99 - SAGA CONTINUES

by: Michal Jaegermann

First news! Announced in a previous instalment an update of c99 compiler is here. Check our software library. It improves and expands the compiler. In particular 2-dimensional arrays are now directly supported. You will find also recoded and enriched i/o library (printf and scanf family), with some documentation. Other stuff. All in all - one full SSSD disk of software. All of this straight from the author, Clint Pulley.

Thank you, Clint!

Undoubtly you have noticed, in the first article, traces of "the formatter strikes again" syndrome. Bear with us. Things should improve with time.

Finally, let me tell you - hope that will be not censored away - that our great editor sometimes gets carried away in his duties. That means that I am finding, from time to time, signed with my name, things of which I never dreamed of. Just to set a record straight. C is not like a grand-daddy Fortran. I should know that, since I earn for my upkeep writing, among other things, some Fortran programs (sigh...). You wouldn't want to learn all weird things which can happen to you in a True Blue Fortran Compiler. In a broad sense, C, together with Pascal, belongs to a family of post-Algol, structured, procedural programming languages. This is a world apart from our venerable, but a little bit dated, precedessor. If you do not understand that lingo, do not worry. In a comparison with BASIC, nearly everything is structured. Next, C directive #define, or assembler EQU are like Pascal declaration "const", but never like LET statement in BASIC. LET is just a marker for an assignment statement, which uses, in C, an "=" sign. #define (with a "hash" always in the first column) lets you define a name for a constant used in a program. For example, if you will write "#define RED 7", you may write later, in c99 program, "screen (RED);", which has effects similar to BASIC's "CALL SCREEN(7)", but is quite a bit more readable. Also, if you have in your program twenty occurrences of "screen (RED);", and decide later that the other kind of red is better, you are just changing one "define" to, say, "#define RED 10", instead of hunting for every "7" and wondering which one you missed. Upper case for a constant name is a pure stylistical convention, just to mark that you are not dealing with a variable. Last, but not the least, arguments to C, and c99 as well, functions are passed similar to parenthesized parameters of X-BASIC subprograms, and not subroutines. I stress very strongly "parenthesized", since that means that you are passing only VALUES and not variable NAMES. This is much under-utilized feature of X-BASIC, very far cry from a "standard" BASIC, and if you forgot what is a difference, please, check your manual.

After this longish introduction let's do some real work. I devoted over three pages of the last newsletter explaining how to print a digit on a screen. So how about a whole number for a change? Lets print a positive integer, right justified in a field of a width six, filled with blanks. Exciting, isn't it? Hold on. More exciting than you suspect right now.

"Bare bones" c99 provides basic functions to handle dislay. Among others a function called puts, which writes a string to a screen. A string in C is any sequence of characters terminated with a NULL (ASCII 0). If you will look how BASIC stores strings, you will find that first byte of a storage area gives a length, thus limiting the longest string to 255 characters. In C a string can be as long as the whole available memory, but it must be terminated with NULL.

The simplest method to print our number is to convert it to string and use puts to print it out. So we need a buffer

for 6 displayable characters plus one extra location for a terminator. To inform the c99 compiler about our request use the following declaration:

```
char buffer [7];
```

"char", because we will be dealing with characters, not "int"egers. Alas, no other types, but those two, are available in c99, though older brothers have many more of them.

DANGER!!! An array buffer has it's cells numbered 0, 1, 2, 3, 4, 5 and 6. There is no cell named "buffer [7]", but if you will try to store something in "buffer [7]", or "buffer [8]" for that matter, C will happily oblige, on an assumption, that as a wise programmer you know better what you are doing. If you are lucky, you will write over non-initialized memory, and you will get away with it. But there is a better chance, that you will over write something important and your program will mysteriously crash. An ever popular bug.

Ok! Now we may prepare the buffer for further work, by filling it with spaces and writing a terminator. It is always a good idea, just in case. You can be mighty surprised with a look of your screen if you will forget about it. This can be done as follows:

```
.................
for (i=0; i<6; i++)
       buffer [i] = ' ';
buffer [6] = NULL;
.................
```

The constant NULL is defined in a file STDIO, which should be "#include"d, or you may write your own "#define".

But there is more then one way to skin a cat. One may use pointers - a subject which we were not touching until now. What is a pointer. Very simple - a location in memory, which stores an address of some other object. Obvious, isn't it? Now repeat the above ten times and everything will be crystal clear (just kiddin'). But still we can probably use some examples, which will show what all of this is about.

Lets have a pointer which points to some location in a 'buffer', which is an array of 'char's. Therefore we will declare the pointer as

```
char *ptc;
```

In translation - a location 'ptc' holds an address of something of a type 'char'. This star - in a declaration - means exactly that. Now lets put something there, like an address of an initial cell of 'buffer'. This may look as follows:

```
ptr = &buffer [0];
```

An expression on right hand side gives exactly that - a required address. This can be written in a shorter form. By a convention, a name of an array, used in an expression, evaluates to an address of an initial cell. So the assignment above can be written as well as:

```
ptc = buffer;
```

and the whole initialization loop may look like that:

```
..............
ptc = buffer;
for (i=0; i<6; i++)
       *ptc++ = ' ';
*ptc = NULL;
..............
```

Now, hold on for a minute. What about this funny "*ptc++" business. We thought that "*" simply denotes a pointer. Well, in a declaration yes. When used in an expression like above, (and when it does not mean MULTIPLY), "*" means: a contents of a location which address is stored in a pointer. A whole statement " *ptc++ = ' '; " describes really two different actions. Take first an address stored in ptc and replace a contents of a location pointed by it by blank (*ptc = ' ';) and next increment the address, so it will point to the next location (ptc++; or equivalent ptc = ptc + 1;). An exercise for you. How to write an expression which will first increment address and later store. And what about "decrement-and-store" (before and after). One more - how to increment a contents of a location pointed to, without touching the pointer. Hint: some parenthesis has to be used, or split it into two expressions.

VERY IMPORTANT!! " ptc++; ", or equivalent " ptc = ptc + 1; ", does not mean "increment an address stored by 1". It means: "change an address stored in ptc in a such manner, that it will point to the next object of a type pointed by ptc". So, it this object happens to be 23 bytes wide, then the address will grow by 23. The last one is not the case with c99, where pointers can point only to objects of type int or char (not even (int *) or (char *)), but in a full C compiler you may create 23-byte wide things, no problem. What is essential, that using pointers you are not concerned with low level details of an implementation. This worries belong to a compiler writer. But you have to take care

that your pointers are properly declared.

At last we are ready to tackle our original problem. Here is the plan:

- set up work buffer
- extract the last digit, which is remainder of a division by 10, and put it in the last available fer location
- replace a value of the argument by the result of an integer division by 10
- repeat the last two steps until an argument value is zero
- print out the whole buffer

and its implementation in c99:

```
dispint (num)
integer num;
{
        char buffer [7], *ptc;
        int i;

        /* clean-up the buffer */
        ptc = buffer;
        for (i=0; i<6; i++)
                *ptc++ = ' ';
        *ptc = NULL;

        /* get consecutive digits */
        do
        {
                *(--ptc) = (num % 10) + '0';
        } while ( num = num / 10 );

        /* print out the whole thing */
        puts (buffer);
}
```

Trace how pointer walks through a buffer - there and back again. The function "puts" requires, as an argument, a pointer to a character, so we supplied one. Namely, an address of the beginning of "buffer". If you feel more comfortable you may write as well " puts (&buffer [0]); ". And something else. One more of opaque C tricks requires an explanation. A way in which I terminated the do-loop. By the way. Do you know why I am checking the condition after the loop and not before?

Every assignment in C has its value. Namely the value of its left hand side - AFTER the job is done. Conditional expressions will consider every non-zero as true and zero as false. So the above means: "replace num by num / 10 and repeat the loop until num is zero". A clearer way to write the same condition would be " while (0 != (num = num / 10)); ". A good optimizing compiler will generate in both cases even the same code. But this is not necessarily true with a simple one-pass compiler like c99.

So everything now is ok, right? Wrong! How about a sign. If you do not know what I mean, write a program which uses the function above and pass -1 to it. What will happen? I will leave as an exercise for you to repair that deficiency. Another exercise - expand this function, so it will accept as its arguments a number and a display base, not necessarily not bigger then 10, and will handle a display properly. And another one - a function which will handle a full range of unsigned integers. A little bit tricky in c99, but possible. Something for ambitious. A function which will display signed and unsigned long integers (2 words of storage). The easiest way probably requires some help from an assembler, but the function is very useful when writing games. With a full range of unsigned longs you may have a score over four billions. And once you are done, repeat all the above for a left justified display. Careful, a display field may contain something else then blanks. An old, bigger value, for example.

A question may arise, why I bother you with all this stuff, whereas we may use, especially with the last update of the compiler, nice and ready functions like "printf". Well, one reason is that this familiar topic lets me to explain many features and constructs of C. And the other one, that all this ramblings may turn out to be much more useful than you think. Library functions from "printf" family, are, by a necessity, of "... but a kitchen sink" kind. Check your documentation for possible options. So they are bulky and comparatively slow. (Everything is blindingly fast in a comparison with BASIC, but this is another story). Very often this is fine. Your time and convenience are more important then extra disk storage and speed is more then acceptable. But when you are writing your ultimate computer game, then all this nifty screens and ten incarnations of that wicked green alien are eating a computer memory rather fast. And you have a lot of other places to burn these extra cycles. Besides of that fancier formats, unusual bases and such are not covered by printf. See exercises above. Recently, in my work, I needed immediately a routine to display bit patterns of arguments. A couple of lines of C has done a trick. Would you like to try your hand? If you think that there is only one way, you are mistaken. If you think that there are only two ways - you are mistaken too. If you will create something you like, write a letter to our editor, put it on the board, bring to the meeting and show to others. Now, who is coming with prizes?

BASICALLY SPEAKING

by: Bob Pass

This month, Jim Beck has contributed three programs for you to type in from your console. The first one is in console basic and it is a game which will require one joystick. Jim calls it "EXTERMINATOR". You have been summoned to the castle of Dr. A. Cula (nice touch there) to exterminate some bats for the doctor. He has left money in each of the rooms which you can have once the room is cleared of the pesky (also deadly) bats. The bats can be exterminated by getting them into the bat-traps in each room. Not too difficult? Well, the good doctor also has a collection of "pets" which like to dine on visitors, so you have to watch for them too. Don't worry about remembering all this as Jim has made the program self instructing. Nice graphics too.

The second program is in extended basic and is an adaptation of one of the programs in an old Miller's Graphics newsletter. It creates rather interesting, ever changing color patterns on the screen. It is an impressive demonstration of how much can be done in just eight lines of code. No documentation is required for this one. Just type it in and run it.

The third program is also in extended basic but you can easilly change it to basic by altering lines one and two to PRINT commands and by splitting line 34 into two successive commands. This latter conversion will also require incrementing the succeeding line numbers by one. This program plays the theme song from Beverly Hills Cops and is very good. Simply type it in and run it; no documentation required.

All three programs can be found listed separately in this newsletter. Type 'em in and enjoy courtesy of Jim!

IBM COMPATIBILITY FOR THE 99/4A

from: TI-BBS

The following is from the January 16'th news release by MG (formerly Miller's Graphics) about their product for IBM compatibility for the TI-99/4A.

Technical Info:

1. Two part system. A TURBO XT and a small bridge box that connects to the side I/O port on your 4A.

2. The TURBO XT is an 8 Mhz/4.77 Mhz (switchable) mother board, power supply, XT style case, CGA color graphics card (both RGB and Composite), Floppy Disk controller, 1 half-high DS/DD disk drive, Parallel port and 256K of Ram on the mother board. The mother board has sockets for up to 640K of ram. There are 8 expansion slots, two of which are used by the CGA card and the Floppy disk controller.

3. The bridge box has inputs for 4A Video in, XT Video in and outputs for XT Keyboard out and Monitor out. It also contains the software for Keyboard switching between 4A mode and XT mode and the software to convert the 4A key strokes into XT keycodes. It also has a pass through so you can keep your P-Box or other Peripherals hooked up.

4. Mode switching from 4A to XT can be done through Basic or X-Basic with CALL XT or by holding down FCTN CTRL ENTER on power up of the 4A.

5. Mode switching from XT to 4A is done by pressing FCTN CTRL ENTER.

6. The ONLY items shared by the two systems are the 4A keyboard and your current monitor or TV. Yes you can get 80 columns out of a composite monitor, but it is easiest to read with the color turned off in 80 mode. The XT allows MODE 40 which also gives you 40 column mode. Graphics programs, such as games and drawing programs work fine in 80 column and most other software that doesn't combine weird foreground and background text colors are also quite readable.

7. By not sharing the disk drives it is possible to do concurrent processing on the XT. Example: Go into XT mode, start up your COMMUNICATIONS software, log on to a BBS and start a down load. Now you can switch modes back to the 4A and do whatever you would like in 4A mode while the XT is still down loading from the BBS!!

8. We have tested this system on a number of 4A system configurations and have found it to be very compatible. Since it is an IBM clone it is also fully compatible with both IBM software and IBM HARDWARE. Yes, you can add ANY IBM cards you would like to the system.

9. The minimum 4A system requirements: TI-99/4A console and a monitor or a TV set with RF modulator.

General Info:

1. This system is being marketed by Triton Products Company in San Francisco, CA. They are also handling the production of the bridge boxes and they have contracted for the Turbo XT clones to their specifications.

2. The system has a 30 DAY money back guarantee and a 1 YEAR parts and labor warranty.

3. The cost for this system (Turbo XT, Bridge box and cables) is 499.00 plus 19.90 for shipping and handling. (US dollars.)

4. Their toll free number for additional info and/or a 6 page 4 color brochure on this system is 800-227-6900, Monday through Friday - 6AM to 6PM and Saturday 9AM to 4PM, Pacific Time. This number is valid in Canada.

5. Delivery is scheduled to start on March 1'st of this year. We have been using this system for awhile now and we are very pleased with its performance. This isn't vaporware, ALL R&D, testing and software is complete and the units are ready for production, so the March time frame is a reality. At last, a MAJOR expansion for the 4A. We hope you are as pleased with this product as you have been with our other products in the past.

As the Triton Brochure says: MAKE THE IBM CONNECTION TO YOUR TI99/4A

----------MG---------

USER GROUP NEWS

by: Bob Pass

Three years. And two months. That is how long I will have been writing this newsletter by this June's issue. Frankly, it has been quite long enough; my creative juices have just about dried up and I am finding it more and more difficult with each passing month to turn out a good product. It is time to turn the job over to someone with fresh ideas, a new outlook, a change for the sake of change.

So. As of this June's issue, I will turn out my last newsletter. By then, one of you will have to take over the presses. Believe me, this is an interesting, rewarding and vital part of this Users' Group. I strongly encourage members to take up where I leave off. I will assist the new editor (or partnership) to get started and I will continue to support the newsletter with articals from time to time. Think about it folks. And let me know if you are interested.

```
100 CALL CLEAR :: CALL INIT :: CALL LOAD(-31788,232):: PRINT "    MULTICOLOR PATTERNS " :: PRINT "    $$$$$$$$$$$$$$$$$$$$" :: PRINT : : : : : : : : : : :: PRINT "    PRESS ANY KEY"
110 CALL KEY(0,K,S):: IF S=0 THEN 110 ELSE CALL SCREEN(2)
120 CALL CLEAR :: FOR T=1 TO 7 :: FOR I=34 TO 126 :: PRINT CHR$(I);:: NEXT I :: NEXT T
130 FOR I=1 TO 2 :: FOR X=1 TO 12 :: CALL HCHAR(X,1,X+33,32):: CALL HCHAR(25-X,1,X+33,32):: NEXT X
140 FOR X=1 TO 16 :: CALL VCHAR(1,X,X+33,24):: CALL VCHAR(1,33-X,X+33,24):: NEXT X :: NEXT I
150 FOR D=34 TO 37 :: FOR X=1 TO 12 :: CALL HCHAR(X,1,D,32):: CALL HCHAR(25-X,1,D,32):: NEXT X :: NEXT D
160 CALL CLEAR :: FOR D=1 TO 25 :: CALL HCHAR(1+INT(24*RND),1+INT(32*RND),33+INT(92*RND)):: NEXT D
170 FOR I=34 TO 126 :: CALL VCHAR(1,1,I,768):: NEXT I :: GOTO 120
180 END
```

```
1 CALL CLEAR
2 DISPLAY AT(10,1):"THEME FR
OM BEVERLY HILLS COP"
3 DISPLAY AT(14,1):"     PRO
GRAM BY JIM BECK"
4 DIM DRUM(16)
5 DIM BASS(64)
6 DIM TUNE(64)
7 DIM HIGH(64)
8 FOR D=1 TO 16
9 READ DRUM(D)
10 NEXT D
11 FOR D=1 TO 64
12 READ BASS(D)
13 NEXT D
14 FOR D=1 TO 64
15 READ TUNE(D)
16 NEXT D
17 FOR D=1 TO 64
18 READ HIGH(D)
19 NEXT D
20 FOR D=1 TO 64
21 T=T+1 :: IF T=17 THEN T=1
22 CALL SOUND(-1000,TUNE(D),
0,BASS(D),30,HIGH(D),30,-5,3
0)
23 NEXT D
24 FOR D=1 TO 64
25 T=T+1 :: IF T=17 THEN T=1
26 CALL SOUND(-1000,TUNE(D),
30,BASS(D),0,HIGH(D),30,-5,D
RUM(T))
27 NEXT D
28 FOR DE=1 TO 3
29 FOR D=1 TO 64
30 T=T+1 :: IF T=17 THEN T=1
31 CALL SOUND(-1000,TUNE(D),
0,BASS(D),0,HIGH(D),30,-5,DR
UM(T))
32 NEXT D
33 FOR D=1 TO 64
34 T=T+1 :: IF T=17 THEN T=1
35 CALL SOUND(-1000,TUNE(D),
0,BASS(D),0,HIGH(D),0,-5,DRU
M(T))
36 NEXT D
37 NEXT DE
38 GOTO 20
39 DATA 10,30,30,30,10,30,30
,30,30,6,6,30,6,30,6,10
40 DATA 175,175,175,30000,17
5,175,175,156,30000,156,131,
131,156,156,175,30000
41 DATA 175,175,175,30000,17
5,175,175,30000,30000,131,13
1,131,156,156,175,175
42 DATA 139,139,139,30000,13
9,139,139,156,30000,156,131,
131,156,156,175,30000
43 DATA 175,175,175,30000,17
5,175,175,30000,30000,175,15
6,30000,131,30000,117,30000
44 DATA 349,349,349,349,415,
415,415,349,30000,349,466,46
6,349,349,311,311,349,349,34
9,349,523,523,523,30000,3000
0
45 DATA 349
46 DATA 554,554,523,523,415,
415,349,349,523,523,698,698,
415,311,30000,311,262,262,31
1,311,349,30000,349,349,349
47 DATA 349,349,349,349,349,
30000,349,311,30000,262,3000
0,233,30000
48 DATA 30000,30000
49 DATA 698,30000,698,30000,
698,831,30000,831,784,30000,
698,30000,622,30000,30000,30
000,698,30000,698,30000,698,
30000
50 DATA 622,698,30000,698,30
000,30000,30000,30000,30000,
30000,554,30000,554,30000,55
4,30000,554,622
51 DATA 30000,622,30000,622,
30000,622,698,30000,698,3000
0,698,30000,698,30000,622,69
8,30000,698,30000
52 DATA 30000,30000,30000
53 END
```

```
10 CALL CLEAR
20 CALL SCREEN(8)
21 CALL CHAR(120,"00000044AA110000")
22 CALL CHAR(121,"000020518A020000")
30 PRINT " EXTERMINATOR  BY JIM BECK"
40 PRINT "xyxyxyxyxyxyxyxyxyxyxyxyxyxyxyxyxy"
50 PRINT
60 PRINT "   YOU ARE AN EXTERMINATOR."
70 PRINT "YOU MAKE YOUR LIVING "
80 PRINT "EXPELLING SPIDERS, ROACHES,"
90 PRINT "AND MICE FROM PEOPLES "
100 PRINT "HOUSES. TODAY, WHILE WORKING"
110 PRINT "THE NIGHT SHIFT, A STRANGE"
120 PRINT "LETTER ARRIVES:"
130 PRINT
140 PRINT "   HELP! COME QUICKLY! MY "
150 PRINT "CASTLE IS BEING TAKEN OVER "
160 PRINT "BY BATS. PLEASE HURRY!"
170 PRINT
180 PRINT "              Dr.A. Cula"
190 PRINT
200 PRINT
210 PRINT
220 PRINT
230 PRINT
240 PRINT " PRESS ANY KEY TO CONTINUE."
250 CALL KEY(0,K,S)
260 IF S=0 THEN 250
270 CALL CLEAR
280 PRINT "   YOU HURRY OVER TO THE"
290 PRINT "PLACE AND KNOCK ON THE DOOR."
300 PRINT "THE HUGE DOOR OPENS AND IN"
310 PRINT "FRONT OF YOU STANDS A SKINNY"
320 PRINT "OLD MAN IN A BLACK CLOAK. HE"
330 PRINT "SAYS THAT YOU MUST GET RID"
340 PRINT "OF THE BATS AND PICK UP YOUR"
350 PRINT "PAY FROM EACH OF THE ROOMS "
360 PRINT "IN THE CASTLE. BUT BEWARE!"
370 PRINT "THIS OLD GUY KEEPS ""PETS"""
380 PRINT "TO. THEY LIVE IN THE DUNGEON"
390 PRINT "BUT THEY LOVE TO EAT PEOPLE."
400 PRINT "   BATS ARE DANGEROUS -NZ "
410 PRINT "CAN BITE YOU. LURE THEM INTO"
420 PRINT "THE BAT-TRAPS TO DESTROY "
430 PRINT "THEM. "
431 PRINT
432 PRINT
433 PRINT
434 PRINT " PRESS ANY KEY TO CONTINUE."
435 CALL KEY(0,K,S)
436 IF S=0 THEN 435
437 CALL CLEAR
440 PRINT "   JUST BEFORE THE OLD MAN"
450 PRINT "LEAVES, HE LOOKS BACK AT YOU"
460 PRINT "AND SAYS:"
470 PRINT "   ""OH, BY THE WAY, I "
480 PRINT "ALMOST FORGOT! YOU MOVE WITH"
490 PRINT "THE JOYSTICK AND PICK UP "
500 PRINT "PAY BY WALKING INTO IT."""
510 PRINT
511 PRINT
512 PRINT
513 PRINT
514 PRINT
515 PRINT
520 PRINT " PRESS ANY KEY TO CONTINUE."
530 CALL KEY(0,K,S)
540 IF S=0 THEN 530
550 CALL CLEAR
1000 RANDOMIZE
1010 CALL CLEAR
1020 DIM RMS(7)
1030 DIM KILL(7)
1040 DIM POF(41)
1050 POF(26)=1
1060 POF(19)=2
1070 POF(18)=3
1080 POF(22)=4
1090 POF(23)=5
1100 POF(28)=6
1110 POF(41)=7
1120 CH1$(1)="387CD6FEC6C6FEAA"
1130 CH1$(2)="387CD6FEFEEEFE54"
1140 CHB$(1)="000042A5181800 00"
1150 CHB$(2)="000000FF180000 00"
1160 CHB$(3)="0000181824C300 00"
1170 CHB$(4)="000000FF180000 00"
1180 DIM BPOS(10,2,7)
1190 R=21
1200 C=12
1210 FOR D=1 TO 7
1220 FOR DE=1 TO 10
1230 BPOS(DE,1,D)=14+INT(8*RND)
1240 BPOS(DE,2,D)=4+INT(24*RND)
1250 NEXT DE
1260 NEXT D
1270 DIM MP(4,2)
1280 CALL CHAR(96,CH1$(1))
1290 CALL CHAR(104,"00DFDFDF00FBFBFB")
1300 CALL CHAR(112,"0200100001400008")
1310 CALL CHAR(113,"0200103811280008")
1320 CALL CHAR(114,"8142241818244281")
1330 CALL CHAR(120,CHB$(1))
1340 CALL CHAR(121,"0200100001400008")
1350 CALL CHAR(122,"9924669999662499")
1360 CALL CHAR(123,"1818003C5A182424")
1370 CALL CHAR(132,"3810102854AAD67C")
1380 CALL COLOR(13,12,8)
1390 CALL CHAR(32,"0200100001400008")
1400 CALL CHAR(35,"55AA55AA55AA55AA")
1410 CALL CLEAR
1411 FOR D=1 TO 7
1412 IF RMS(D)<>2 THEN 1420
1413 NEXT D
1414 MSG$="YOU'VE KILLED ALL THE BATS -NZ COLLECTED YOUR PAY!     -NZ TO GO!! YOU WIN!"
1415 GOTO 6000
1420 CALL SCREEN(2)
1430 FOR DE=9 TO 12
```

```
1440 CALL COLOR(DE,2,2)
1450 NEXT DE
1460 MP(1,1)=4
1470 MP(2,1)=6
1480 MP(3,1)=6
1490 MP(4,1)=4
1500 MP(1,2)=17
1510 MP(2,2)=19
1520 MP(3,2)=23
1530 MP(4,2)=21
1540 FOR D=1 TO 10
1550 PRINT "
          "
1560 NEXT D
1570 PRINT "   hhhhhhhhhhhhh
hhhhhh      "
1580 PRINT "   hyyyyhyyyyhpp
ppppph      "
1590 PRINT "   hyyyyhyyyyhph
phhhphh     "
1600 PRINT "   hyyyyhyyyyhph
ppphpph     "
1610 PRINT "hhhhyyyyhyyyyhph
hhphhph     "
1620 PRINT "hhhhhh#hhh#hhhph
hhphhphhhh  "
1630 PRINT "hyyyyhpppppppppp
pppppppppph "
1640 PRINT "hyyyy#pppppppppp
pppppppppph "
1650 PRINT "hyyyyhpppppppphh#
hhhpppppphh "
1660 PRINT "hyyyyhphhhhhhhyy
yyhhhhhhpph "
1670 PRINT "hhhhhhphyyyyhhyy
yyhyyyyhpph "
1680 PRINT "hhhhhhp#yyyyhhyy
yyhyyyy#pph "
1690 PRINT "hyyyyhphyyyyhhyy
yyhyyyyhpph "
1700 PRINT "hyyyyhphyyyyhhhh
hhhyyyyhpph"
1710 PRINT "hyyyy#phhhhhhhhh
hhhhhhhhpph "
1720 PRINT "hyyyyhpppppppppp
pppppppppph "
1730 PRINT "hhhhhhhhhhhppphhh
hhhhhhhhhh  "
1740 PRINT "       rpppppppp
r           "
1750 PRINT "       rpppppppp
r           "
1760 PRINT "       rrrrrrrrr
r           "
1770 PRINT "
           "
1780 CALL COLOR(9,2,12)
1790 CALL COLOR(10,9,15)
1800 CALL COLOR(11,2,12)
1810 CALL COLOR(12,2,8)
1820 CALL COLOR(1,2,3)
1830 CALL SCREEN(3)
1840 CALL HCHAR(R,C,113)
1850 FOR D=1 TO 4
1860 CALL HCHAR(MP(D,1),MP(D
,2),96)
1870 NEXT D
1880 CALL JOYST(1,X,Y)
1890 X1=-Y/4
1900 Y1=X/4
1910 IF X1=0 THEN 1940
1920 Y1=0
1930 GOTO 2000
1940 IF Y1<>0 THEN 2000
1950 T=T+1
1960 IF T<3 THEN 1980
1970 T=1
1980 CALL CHAR(96,CH1$(T))
1990 GOTO 1880
2000 CALL GCHAR(R+X1,C+Y1,FR
)
2010 IF FR=96 THEN 2130
2020 IF FR=35 THEN 2370
2030 IF FR<>112 THEN 1880
2040 R=R+X1
2050 C=C+Y1
2060 CALL HCHAR(R,C,113)
2070 CALL HCHAR(R-X1,C-Y1,11
2)
2080 FOR D=1 TO 4
2090 IF MP(D,2)<C THEN 2250
2100 IF MP(D,2)>C THEN 2280
2110 IF MP(D,1)<R THEN 2310
2120 IF MP(D,1)>R THEN 2340
2130 MSG$="OH MY! ONE OF THE
 MONSTERS  JUST HAD LUNCH. Y
OU LOSE!"
2131 GOTO 6000
2140 CALL GCHAR(MP(D,1)+MX,M
P(D,2)+MY,FT)
2150 IF FT=113 THEN 2130
2160 IF FT=112 THEN 2190
2170 IF MY<>0 THEN 2110
2180 GOTO 2230
2190 CALL HCHAR(MP(D,1),MP(D
,2),112)
2200 MP(D,2)=MP(D,2)+MY
2210 MP(D,1)=MP(D,1)+MX
2220 CALL HCHAR(MP(D,1),MP(D
,2),96)
2230 NEXT D
2240 GOTO 1880
2250 MX=0
2260 MY=1
2270 GOTO 2140
2280 MX=0
2290 MY=-1
2300 GOTO 2140
2310 MX=1
2320 MY=0
2330 GOTO 2140
2340 MX=-1
2350 MY=0
2360 GOTO 2140
2370 CALL CLEAR
2380 CALL COLOR(1,2,2)
2390 CALL SCREEN(2)
2400 CALL COLOR(10,2,2)
2410 CALL COLOR(12,2,2)
2420 CALL HCHAR(1,1,121,786)
2430 CALL HCHAR(1,1,104,32)
2440 CALL HCHAR(24,1,104,32)
2450 CALL VCHAR(1,1,104,24)
2460 CALL VCHAR(1,32,104,24)
2470 PO=POF(R+C)
2480 IF RMS(PO)=2 THEN 2590
2490 IF RMS(PO)=0.5 THEN 258
0
2500 FOR D=1 TO 70
2510 CALL HCHAR(2+INT(20*RND
),2+INT(28*RND ),122)
2520 NEXT D
2530 FOR D=1 TO 10
2540 IF BPOS(D,1,PO)=100 THE
N 2560
2550 CALL HCHAR(BPOS(D,1,PO)
,BPOS(D,2,PO),120)
2560 NEXT D
2570 IF RMS(PO)=1.5 THEN 259
0
2580 CALL HCHAR(3,3,132)
2590 CALL COLOR(12,2,8)
2600 CALL COLOR(12,2,8)
2610 CALL COLOR(1,2,3)
2620 CALL COLOR(10,7,15)
2630 CALL HCHAR(12,16,121)
2640 IF X1=-1 THEN 3400
2650 IF X1=1 THEN 3350
2660 IF Y1=-1 THEN 3500
2670 IF Y1=1 THEN 3450
2680 RB=RB+X1
2690 CB=CB+Y1
2700 CALL GCHAR(RB,CB,FR)
2710 IF FR<>121 THEN 3650
2720 CALL HCHAR(RB-X1,CB-Y1,
121)
```

ON THE FASTRACK


```
2730 CALL HCHAR(RB,CB,123)
2740 IF RB=12 THEN 2760
2750 GOTO 2680
2760 IF CB=16 THEN 2780
2770 GOTO 2680
2780 CALL JOYST(1,X,Y)
2790 X1=-Y/4
2800 Y1=X/4
2810 IF X1<>0 THEN 2880
2820 IF Y1<>0 THEN 2880
2830 BAT=BAT+1
2840 IF BAT<5 THEN 2860
2850 BAT=1
2860 CALL CHAR(120,CHB$(BAT)
)
2870 GOTO 2780
2880 CALL GCHAR(RB+X1,CB+Y1,
FR)
2890 IF FR<>35 THEN 2920
2900 CALL COLOR(1,2,2)
2910 GOTO 1410
2920 IF FR<>132 THEN 2980
2930 FOR SOU=1 TO 15
2940 CALL SOUND(-200,990,SOU
)
2950 NEXT SOU
2960 RMS(PO)=RMS(PO)+1.5
2970 GOTO 2990
2980 IF FR<>121 THEN 2830
2990 RB=RB+X1
3000 CB=CB+Y1
3010 CALL HCHAR(RB-X1,CB-Y1,
121)
3020 CALL HCHAR(RB,CB,123)
3030 FOR D=1 TO 10
3040 BX1=BPOS(D,1,PO)
3050 BY1=BPOS(D,2,PO)
3060 IF BX1=100 THEN 3320
3070 IF BX1<RB THEN 3550
3080 IF BX1>RB THEN 3570
3090 IF BY1<CB THEN 3590
3100 IF BY1>CB THEN 3610
3110 GOTO 3130
3120 MSG$="OUCH!!! GOT BIT B
Y A BAT!   YOU LOSE!"
3121 GOTO 6000
3130 CALL GCHAR(BX1,BY1,FR)
3140 IF FR<>122 THEN 3260
3150 CALL HCHAR(BPOS(D,1,PO)
,BPOS(D,2,PO),121)
3160 FOR DEA=1 TO 5
3170 CALL HCHAR(BX1,BY1,122)
3180 CALL HCHAR(BX1,BY1,120)
3190 CALL SOUND(-300,-7,0)
3200 NEXT DEA
3210 KILL(PO)=KILL(PO)+1
3220 IF KILL(PO)=10 THEN 363
0
3230 CALL HCHAR(BX1,BY1,121)
3240 BPOS(D,1,PO)=100
3250 GOTO 3320
3260 IF FR=123 THEN 3120
3270 IF FR=120 THEN 3150
3280 CALL HCHAR(BPOS(D,1,PO)
,BPOS(D,2,PO),121)
3290 CALL HCHAR(BX1,BY1,120)
3300 BPOS(D,1,PO)=BX1
3310 BPOS(D,2,PO)=BY1
3320 NEXT D
3330 GOTO 2780
3340 GOTO 3340
3350 CALL HCHAR(1,15,35,3)
3360 RB=2
3370 CB=16
3380 CALL HCHAR(2,16,123)
3390 GOTO 2680
3400 CALL HCHAR(24,15,35,3)
3410 RB=23
3420 CB=16
3430 CALL HCHAR(23,16,123)
3440 GOTO 2680
3450 CALL VCHAR(11,1,35,3)
3460 RB=12
3470 CB=2
3480 CALL HCHAR(12,2,123)
3490 GOTO 2680
3500 CALL VCHAR(11,32,35,3)
3510 RB=12
3520 CB=31
3530 CALL HCHAR(12,31,123)
3540 GOTO 2680
3550 BX1=BX1+1
3560 GOTO 3090
3570 BX1=BX1-1
3580 GOTO 3090
3590 BY1=BY1+1
3600 GOTO 3130
3610 BY1=BY1-1
3620 GOTO 3130
3630 RMS(PO)=RMS(PO)+0.5
3640 GOTO 3230
3650 CALL HCHAR(RB-X1,CB-Y1,
121)
3660 RB=RB+X1
3670 CB=CB+Y1
3680 IF RB<>12 THEN 2740
3690 IF CB<>16 THEN 2740
3700 GOTO 2730
6000 CALL CLEAR
6001 CALL CHAR(32,"")
6002 CALL SCREEN(2)
6010 PRINT MSG$
6020 PRINT
6030 PRINT
6040 PRINT
6050 PRINT
6060 PRINT
6070 PRINT
6080 PRINT "   PRESS ANY KE
Y TO END.  "
6081 FOR D=1 TO 14
6082 CALL COLOR(D,16,2)
6083 NEXT D
6090 CALL KEY(0,K,S)
6100 IF S=0 THEN 6090
6110 CALL CLEAR
6120 END
```