C.C.O.G.

COLOR COMPUTER OWNERS GROUP

NEWSLETTER JULY 1991

INDEX:

PRESIDENTS COMMENTS:

Wow, do you believe 20 pages of articles. Robert and Tom got a little carried away this month. But, I think its great!!! It may be a little too technical for some but we try to have a little for everyone. So from time to time we should be allowed to go a little overboard one way or another.

One of the advantages of writing this column and editing the newsletter is that I get to read all of the other columns before I write this one. That does help me sometimes with a thought or two when writing this column.

I don't know what article Gus read about the Apple/IBM joint venture. I have seen a lot of information and a lot of mis-information since this venture was announced. I will give you my version of what I have been able to get out of what I have read. The machine will be built around the IBM RISC-6000 chip. RISC = Reduced Instruction Set Chip. RISC chips are used in Workstations and MINI-computers. They are very fast and very expensive. At least up to now they have been expensive. The machine will have a GUI, Graphical User Interface, built on top of IBM's operating system. The machine will be aimed at high-end CAD users, and at commercial and industrial Multi-media markets. The machine will be rather expensive when first released.

This venture does two things for IBM: 1. A new market for their RISC-6000 chip. And 2. It get Apple off their backs about GUI's (WINDOWS).

This venture does two things for Apple: 1. It gives them a machine to compete with NeXT, Appolo, and Sparc. And 2. It gives them an upward path for Macintosh users. Rumor has it that the supply of CPU's for The Mac is in jeopardy because of a lawsuit filed against Motorola by a Japanese firm.

It will be a long time (2 years after release) before a machine of this type will have any impact on the computer hobbyist. Bernie

COLOR COMPUTER OWNERS GROUP

**This newsletter is a periodic publication of the
COLOR COMPUTER OWNERS GROUP
of Metropolitan Detroit.**

Contact: Bernie Patton (313) 283-2474 BBS (313) 292-4713

C.C.O.G. Minutes from June 25, 1991 by Marcine Glowicki

The meeting was called to order at 7:18. 18 people were in attendance. A motion was made to accept the minutes from last month's meeting. Motion passed.

OLD BUSINESS--An invitation was made to accept advertising from all advertisers who are still in business and promoting the COCO machine. We will reserve the right to shrink the size of the advertisement submitted but we will not edit the advertisement.

While people were airing their opinions tonight, someone said we should contact Kentucky (RAINBOW MAGAZINE), and complain about the size of the magazine and the lack of material on the disk. Bernie stepped in on this one and referred to his article in the June newsletter.

Frank Hogg was approached by Tandy to purchase ROMS--no word yet.

NEW BUSINESS--Robert Gault informed members that the old gimmi chip can cause computer problems of various sorts. If the chip has a 1986 date this is an older version of the chip. The 1987/88 dated chip is the improved version. Bernie received a letter from Tenn. asking if the club had any information to share with regards to the new MMI computer. Specifically they wanted comments from anyone who had purchased one of the machines. This information will be compiled into a book that the fellow is compiling on the MMI.

Demonstrations for the evening went into a presentation of RS Basic by Bernie, Bob Gault on OS9 basic and Tom with C. It was interesting that they each worked with the same type program but showed how it could be done using each of the types of programing.

Next month's theme will be discussed on the BBS so voice your opinion and Bernie will take count. Actually this might be a way to try and get more people from the club to use the BBS.

$17 was collected for the VFW.

See you next month, July 23, 1991.

OS-9 by Tom Napolitano

This article contains minimal answers to the comp.lang.c frequently-asked questions list.

Null Pointers

Q. What is this infamous null pointer, anyway?
A: For each pointer type, there is a special value -- the "null pointer" -- which is distinguishable from all other pointer values and which is not the address of any object.

Q. How do I "get" a null pointer in my programs?
A: A constant 0 in a pointer context is converted into a null pointer at compile time. A "pointer context" is an initialization, assignment, or comparison with one side a variable or expression of pointer type, and (in ANSI standard C) a function argument which has a prototype in scope declaring a certain parameter as being of pointer type. In other contexts (function arguments without prototypes, or in the variable part of variadic function calls) a constant 0 with an appropriate explicit cast is required.

Q. How should NULL be #defined on a machine which uses a nonzero bit pattern as the internal representation of a null pointer?
A: The same as any other machine: as 0 (or (void *)0). (The compiler makes the translation, upon seeing a 0, not the preprocessor.)

Q. Is the abbreviated pointer comparison "if(p)" to test for non-null pointers valid? What if the internal representation for null pointers is nonzero?
A: The construction "if(p)" works, regardless of the internal representation of null pointers, because the compiler essentially rewrites it as "if(p != 0)" and goes on to convert 0 into the correct null pointer.

Q. If "NULL" and "0" are equivalent, which should I use?
A: Either; the distinction is entirely stylistic.

Q. But wouldn't it be better to use NULL (rather than 0) in case the value of NULL changes, perhaps on a machine with nonzero null pointers?
A: No. NULL is, and will always be, 0.

Q. Why is there so much confusion surrounding null pointers? Why do these questions come up so often?
A: The fact that null pointers are represented both in source code, and internally to most machines, as zero invites unwarranted assumptions. The use of a preprocessor macro (NULL) suggests that the value might change later, or on some weird machine.

Q. I'm still confused. I just can't understand all this null pointer stuff.
A: A simple rule is, "Always use '0' or 'NULL' for null pointers, and always cast them when they are used as arguments in function calls."

Arrays and Pointers

Q. I had the definition char x[6] in one source file, and in another I declared extern char *x. Why didn't it work?
A: The declaration extern char *x simply does not match the actual definition. Use extern char x[].

Q. But I heard that char x[] was identical to char *x.
A: Not at all. Arrays are not pointers.

Q. Why are array and pointer declarations interchangeable as function formal parameters?
A: Since functions can never receive arrays as parameters, any parameter declarations which "look like" arrays are treated by the compiler as if they were pointers.

Q. Someone explained to me that arrays were really just constant pointers.
A: An array name is "constant" in that it cannot be assigned to, but an array is _not_ a pointer.

Q. I came across some "joke" code containing the "expression" 5["abcdef"] . How can this be legal C?
A: Yes, array subscripting is commutative in C. The array subscripting operation a[e] is defined as being equivalent to *((a)+(e)).

Q. How do I declare a pointer to an array?
A: Usually, you don't want to. Consider using a pointer to one of the array's elements instead.

Q. How can I dynamically allocate a multidimensional array?
A: It is usually best to allocate an array of pointers, and then initialize each pointer to a dynamically-allocated "row." See the full list for code samples.

Order of Evaluation

Q. Under my compiler, the code "int i = 7; printf("%d\n", i++ * i++);" prints 49. Regardless of the order of evaluation, shouldn't it print 56?
A: The operations implied by the postincrement and postdecrement operators ++ and -- are performed at some time after the operand's former values are yielded and before the end of the expression, but not necessarily immediately after, or before other parts of the expression are evaluated.

Q. But what about the &&, ||, and comma operators?
A: There is a special exception for those operators, (as well as ?:); left-to-right evaluation is guaranteed.

Memory Allocation

Q. Why doesn't the code "char *answer; gets(answer);" work?
A: The pointer variable "answer" has not been set to point to any valid storage. The simplest way to correct this fragment is to use a local

array, instead of a pointer.

Q. I can't get strcat to work. I tried "char *s1 = "Hello, ", *s2 = "world!", *s3 = strcat(s1, s2);" but I got strange results.
A: Again, the problem is that space for the concatenated result is not properly allocated.

Q. But the manual page for strcat says that it takes two char *'s as arguments. How am I supposed to know to allocate things?
A: In general, when using pointers you _always_ have to consider memory allocation, at least to make sure that the compiler is doing it for you.

Q. Can you use dynamically-allocated memory after you free it?
A: No.

Q. How does free() know how many bytes to free?
A: The malloc/free package remembers the size of each block it allocates and returns.

Q. Is it safe to use calloc's zero-fill guarantee for pointer and floating-point values?
A: calloc(m, n) is essentially equivalent to "p = malloc(m * n); memset(p, 0, m * n); ". The zero fill is all-bits-zero, and does not therefore guarantee useful zero values for pointers or floating-point values.

Structures

Q. I heard that structures could be assigned to variables and passed to and from functions, but K&R I says not.
A: These operations are supported by all modern compilers. But unfortunately, not the coco os9 c compiler.

Q. Why can't you compare structs?
A: There is no reasonable way for a compiler to implement struct comparison which is consistent with C's low-level flavor.

Declarations

Q. How do you decide which integer type to use?
A: If you might need large values, use long. If space is very important, use short. Otherwise, use int.

Q. How do I declare an array of pointers to functions returning pointers to functions returning pointers to characters?
A: char *(*(*a[5])())();

Q. How do I initialize a pointer to a function?
A: Use something like "extern int func(); int (*fp)() = func; " .

Boolean Expressions and Variables

Q. What is the right type to use for boolean values in C?
A: C does not provide a standard boolean type, because picking one

involves a space/time tradeoff which is best decided by the programmer.

Q. Isn't #defining TRUE to be 1 dangerous, since any nonzero value is considered "true" in C? What if a built-in boolean or relational operator "returns" something other than 1?
A: It is true (sic) that any nonzero value is considered true in C, but this applies only "on input", i.e. where a boolean value is expected. When a boolean value is generated by a built-in operator, it is guaranteed to be 1 or 0. (This is _not_ true for some library routines such as isalpha.)

Operating System Dependencies

Q. How can I read a single character from the keyboard without waiting for a newline?
A: Contrary to popular belief and many people's wishes, this is not a C-related question. How to do so is a function of the operating system in use.

Stdio

Q. My program's prompts and intermediate output don't always show up on the screen, especially when I pipe the output through another program.
A: It is best to use an explicit fflush(stdout) whenever output should definitely be visible.

Q. When I read from the keyboard with scanf(), it seems to hang until I type one extra line of input.
A: scanf() was designed for free-format input, which is seldom what you want when reading from the keyboard.

Q. How can I recover the file name given an open file descriptor?
A: This problem is, in general, insoluble. It is best to remember the names of open files yourself.

Miscellaneous

Q. What can I safely assume about the initial values of variables which are not explicitly initialized?
A: Variables with "static" duration start out as 0, as if the programmer had initialized them. Variables with "automatic" duration, and dynamically-allocated memory, start out containing garbage (with the exception of calloc).

Q. Can someone tell me how to write itoa?
A: Just use sprintf.

Q. How can I write data files which can be read on other machines with different data formats?
A: The best solution is to use text files.

Q. I seem to be missing the system header file <sgtty.h>. Can someone send me a copy?
A: You cannot just pick up a copy of someone else's header file and

expect it to work, since the definitions within header files are frequently system-dependent.

Q. How can I call Fortran (BASIC, Pascal, ADA, lisp) functions from C?
A: The answer is entirely dependent on the machine and the specific calling sequences of the various compilers in use.

Q. How can I make this code more efficient?
A: Efficiency is not important nearly as often as people tend to think it is. Most of the time, by simply paying attention to good algorithm choices, perfectly acceptable results can be achieved.

Q. Are pointers really faster than arrays? Do function calls really slow things down?
A: Precise answers to these and many similar questions depend of course on the processor and compiler in use.

Q. I'm having trouble with a Turbo C program which crashes and says something like "floating point not loaded."
A: Some compilers for small machines, including Turbo C, attempt to leave out floating point support if it looks like it will not be needed. The programmer must occasionally insert a dummy explicit floating-point call to force loading of floating-point support.

Q. This program crashes before it even runs!
A: Look for very large, local arrays.

Q. How do you pronounce "char"?
A: Like the English words "char," "care," or "car" (your choice).

Abbridged from a text file by:

Steve Summit

"Viewpoints" by Gus Korte"

Recently I completed a program in RSDOS that enhances SCRIPSIT word processing software so that I can use it do bold type printing. Between the helpful suggestions received from CCOG club members on the club electronic bulletin board system (BBS) and what I could extract from my printer manuals, I was able to eventually run the program successfully. I am only an amateur at programming. However the program works and if I can do this so can you other hobbyists who are "green" at programming. Try writing some programs to accomplish processing data for your own purposes. I think you'll enjoy it. As you no doubt know, MAX10 software can already do bold and other fancy printing but the printing is at a much slower rate. So I use SCRIPSIT for my rapid printing jobs and include the bold print option for the more important output. However for the really fancy printing, usually for master copies from which to duplicate more copies, I use MAX10. If any of you who own SCRIPSIT are interested in bold printing your results, let me know and I can make my program available to you.

One of the good reasons to contact the forum messages on the electronic bulletin board system (BBS) the CCOG club supports, is to keep yourself current on the latest information about our hobby. Recently I noticed a message on the BBS about a FD502 disk drive on sale at a nearby Radio Shack store (located near 5 Mile and Merriman Roads in Livonia) for a very low price. I quickly went over there and bought that bargain before it was gone. Now I have to determine how to hook it to my COCO3 through a multipak or other type of system when one becomes available. This again indicates that Radio Shack has some good sale bargains available at this time for all you COCO users. This Radio Shack store, mentioned previously, also has a lot of COCO software and manuals at sale prices so you may be interested in seeing what they have to enhance your hobby. Having a multiple disk drive system is a necessity to get full use of the BASIC09 software and related products, as club member Bob Gault has indicated previously.

I have learned a lot about BASIC09 from reading the manual on the subject which I also obtained at a sale bargain price at Radio Shack. As I get into more of this subject, I can see how much better this type of BASIC is than RSDOS BASIC. That's why many club members feel that OS9 is the main future for all COCO users. So if you can afford to go this route do so since it will make your COCO hobby more enjoyable.

Have you noticed the newspaper report that the Apple and IBM computer companies have decided to cooperate in making simpler-to-use computers? They also want to standardize them too as a cooperative effort. I assume this will also include the computer software languages. I wonder how this will relate to the so-called COCO4 type of computers now developing and even the OS9 system. Does this mean that all future computers will become IBM-Apple compatible in

order to compete? Only time will tell. However I think that as long as our COCO's are performing according to our desires then we don't have to be concerned about these new developments.
Our COCO's will be around for a long time for our use.

A lot of talk has been on the electronic bulletin board system (BBS) that our CCOG club supports regarding the possible use of VCR video taping of instructions about our COCO's. Our club meetings are informative concerning our hobby. Possibly if those parts of the club meeting that instruct us about the COCO are VCR taped, they would be useful especially to those who can't attend every club meeting. Such VCR cassettes could be kept in the club library for members use. One person on the BBS has indicated that he might be willing to buy such a VCR cassette. So that indicates there is an interest there in this type of an activity. How far that will go is not clear now. Perhaps that would be a good topic for discussion at a future CCOG club meeting. Club member Karl Sefcik has indicated to me that he supports the idea if the club decides to try it. Karl Sefcik's message in last month's club newsletter is repeated here for those of you who wish to subscribe to the Rainbow magazine. You can use a free telephone number to order by credit card. It is 1-800-847-0309. Since some of you do not use the BBS yet, this information is included here once again for your information.

A reminder especially for new members of the CCOG, the VFW uses your money donations from our club in it's many charitable activities especially concerning unfortunate hospitalized veterans. It's all going for a good cause. The Livonia VFW also collects used books and magazines to give hospitalized veterans. So if any of you want to contribute such, which you would otherwise discard, give them to the VFW bartender to make some unfortunate veterans happy. Also we owe some thanks to the VFW bartender because he volunteers his time to open the VFW Post for us and serve us drinks. He gets nothing for providing us this service so let him know you appreciate his efforts for us whenever you can.

BEGINNER'S CORNER by Robert Gault

Last month we had a demonstration of the same program in RSDOS, Basic09, and C at the CCOG meeting. The response was underwhelming to say the least. One good question was asked, "how does the speed of the program change from language to language?"

Here is an example of a program in RSDOS, Basic09, and assembly code. Tom may add a C version in his column. The code is designed to time itself and nothing more. While I have made use of a hardware clock in RSDOS, you can run the same program without the date$ call and use a stopwatch.

The results clearly indicate an increase in speed from RSDOS to Basic09 to assembly. Check the code below for actual times.

RSDOS Basic program

```
10 PRINT DATE$
20 FOR I=1 TO 10000:NEXT
30 PRINT DATE$
40 ' ABOUT 9 SEC. AT 2MHZ
```

Basic09 programs

```
PROCEDURE loop
 0000       PRINT DATE$
 0003       FOR i=1 TO 10000
 0016       NEXT i
 0021       PRINT DATE$
 0024       (* Normally 6 sec. at 2MHz


PROCEDURE loop2
 0000       DIM i:INTEGER
 0007       PRINT DATE$
 000A       FOR i=1 TO 10000
 001B       NEXT i
 0026       PRINT DATE$
 0029       (* Normally takes 1 sec. at 2MHz


PROCEDURE loop3
 0000       PRINT DATE$
 0003       RUN mlloop
 0007       PRINT DATE$
 000A       (* 3 sec. at 2MHz for 655,350 counts
```

BEGINNER'S CORNER by Robert Gault Continued

```
Assembly program "mlloop" called by the above Basic09 program
Microware OS-9 Assembler RS Version 01.00.00    06/28/91 15:42:47
Page 001
"mlloop" - "demo for CCOG basic09"
00001                                nam    "mlloop"
00002                                ttl    "demo for CCOG basic09"
00003    0000 87CD0024              mod    endprg, name, langtype, attrev,
                                            start, enddat
00004    000D 6D6C6C6F   name       fcs    "mlloop"
00005    0021           langtype set    $20+1    subroutine; 6809 object
00006    0081            attrev   set    $80+1    reentrant
00007 D  0000            enddat   equ    .
00008    0013 01                  fcb    1
00009    0014            start    equ    *
00010    0014 C60A                ldb    #10
00011    0016 8EFFFF     loop1    ldx    #$ffff      same as 65535
00012    0019 301F       loop2    leax   -1,x        subtract 1 from x
00013    001B 26FC                bne    loop2       branch if not  zero
00014    001D 5A                  decb
00015    001E 26F6                bne    loop1
00016    0020 39                  rts                return to basic09
00017    0021 36CD1B              emod
00018    0024            endprg   equ    *
00019          * net effect of the two loops is to count to 655,350
00000 error(s)
00000 warning(s)
$0024 00036 program bytes generated
$0000 00000 data bytes allocated
$00AC 00172 bytes used for symbols
```

EXPERTS' CORNER
by
Robert Gault

At our last meeting, we compared three versions of the same program in RSDOS, Basic09, and C. The program required random numbers but the Tandy C does not include a random number generator as part of the C library. Tom Napolitano wrote his own random number generator and thus we have the subject of this months column. What is a random number generator, how do you test one for randomness, and how do you write one?

Random means unpredictable and that is the type of number a random number generator should produce. Of course it is not possible for software to be unpredictable but we can settle for a long list of numbers well scrambled. This type is called a pseudo random number generator (prng).

So a good prng will have the longest sequence of numbers possible before the sequence repeats. There should also not be any patterns within the sequence. An example of a bad prng would be one yielding the sequence 0, 1, 2, 3, ... 1,000,000.

Rigorous testing of prngs is a subject beyond my capabilities. Still there are some simple ways to test a prng which can be done on the Coco. The results are good enough to insure any prng that we use for the Coco. So let's start by testing the quality of the prngs supplied with RSDOS and Basic09.

We don't know the sequence length for the Tandy prngs nor the algorithm for Basic09. The algorithm for RSDOS is listed in the "Basic Unravelled" books:
for X(=rnd(1 (1) rnd=a*seed+c (2) seed=low order bytes of rnd (3) scramble lowest order byte of rnd and return value.

I am leery of this algorithm because of steps (2) & (3). Even if the output of a prng is good, the randomness of the bits within any number may not be good. So let's test it. First we will graph random numbers from 1 to 255 and see if any are favored/disfavored.

RANDOM NUMBER 1-255

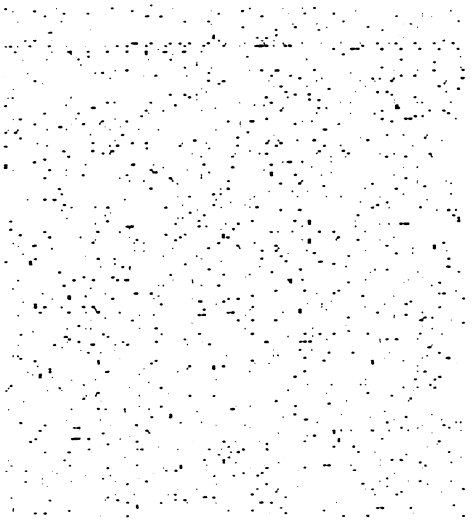
```
10 PMODE4,1:PCLS:SCREEN1,1
20 DIM R(255)
30 FOR I=1 TO 255:R(I)=0:NEXT
40 FOR I=0 TO 1 STEP0
50 R=RND(255):R(R)=R(R)+1
60 PSET(R,R(R))
70 NEXT
```

Ideally we would need need a graph the length of the prng sequence for the above test to indicate goodness by the absence of any gaps. Above, all numbers from 1 to 255 were produced without noticeable favoritism. The fringe varied with time as a random noise.

Our next tests are designed to see if there are any noticeable patterns within the above sequence. We should examine groups of numbers - within the prng sequence - of length 2-6 but we will only look at the pairs and triplets since these can be graphed.

RANDOM PAIRS; RANGE 1-255

```
10 PMODE 4,1: PCLS: SCREEN1,1
12 A=RND(-1): A=RND(0)
20 FOR I=0 TO 1 STEP 0
30 R=RND(0)
40 X=192*R: Y=191*A: A=R
50 PSET(X,Y): NEXT
```

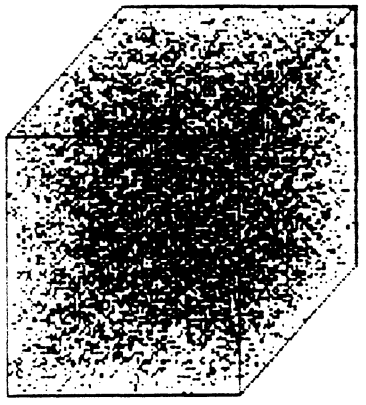


If there are no patterns in number pairs, the above picture will gradually fill until the entire screen is covered. The above pattern looks good for pairs.

RANDOM TRIPLETS; RANGE 1-255

```
10 PMODE4,1: PCLS: SCREEN1,1
20 FOR I=0 TO 96 STEP96
30 LINE(48+I,72)-(96+I,24),PSET
40 LINE(48+I,72)-(48+I,168),PSET
50 LINE(48,72+I)-(144,72+I),PSET
60 NEXT
70 LINE(96,24)-(192,24),PSET
80 LINE(192,24)-(192,120),PSET
90 LINE(144,168)-(192,120),PSET
100 A=RND(-1): B=RND(0): A=RND(0)
```

```
110 FOR I=0 TO 1 STEP0
120 R=RND(0)
130 Z=48*B
140 X=96*R+Z+48: Y=96*A-Z+72
150 B=A: A=R
160 PSET(X,Y)
170 NEXT
```



If there are no patterns in the triplets, the above picture will look like a fuzzy ball contained within the cube. The cube will eventually fill completely. The above pattern seems good for triplets.

While the above is not rigorous testing, RND in RSDOS seems quite usable.

Now let's look at the prng in Basic09. Equivalent programs will be used. The programs are presented below without comment. There are no diagrams because of the difficulty in saving and converting OS-9 diags. to RSDOS. The results are essentially identical to the pictures above. Try the programs and see.

```
PROCEDURE prng1
DIM r(255):INTEGER
DIM i,n:INTEGER
RUN gfx("mode",0,5)
RUN gfx("clear")
FOR i=1 TO 255
   r(i):=0
NEXT i
n:=RND(-1)
LOOP
   n:=RND(254)+1
   r(n):=r(n)+1
   RUN gfx("point",n,r(n))
ENDLOOP
```

```
PROCEDURE prng2
RUN gfx("mode",0,5)
RUN gfx("clear")
n:=RND(-1)
n:=RND(0)
LOOP
   y:=191.*n
   n:=RND(0)
   x:=191.*n
   RUN gfx("point",FIX(x),FIX(y))
ENDLOOP

PROCEDURE prng3
DIM i:INTEGER
RUN gfx("mode",0,5)
RUN gfx("clear")
FOR i=0 TO 96 STEP 96
   RUN gfx("line",48+i,120,96+i,168)
   RUN gfx("line",48+i,24,48+i,120)
   RUN gfx("line",48,24+i,144,24+i)
NEXT i
RUN gfx("line",96,168,192,168)
RUN gfx("line",192,72,192,168)
RUN gfx("line",144,24,192,72)
a:=RND(-1)
b:=RND(0)
a:=RND(0)
LOOP
   r:=RND(0)
   z:=48.*b
   x:=96.*r+z+48.
   y:=96.*a+z+24.
   RUN gfx("point",FIX(x),FIX(y))
   b:=a
   a:=r
ENDLOOP
```

So it looks as though the prngs of RSDOS and Basic09 are reasonable but what to do with C or assembly programs? Clearly we need to write our own prngs and then evaluate them.

Let's try out Tom's prng as an example. Keep in mind that it was a rough and ready effort for a club demo. So if it turns out badly, it is no reflection on Tom as a programmer. The routine in pseudo code is:

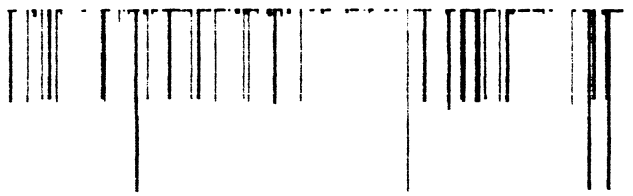
```
srandom:
initialize by  seed = system second*hour*year    last = system minute
```

```
random:
last = last*seed
rnd = mod[abs(last + little),range] where range = big - little +1
                                        big = 100;  little = 0
```

There is a problem with this algorithm which you can see more easily in pseudo code or Basic than in C. It is possible for "last" and/or "seed" to be zero. If that happens, then the prng stays stuck at zero.

I modified the code to prevent zero as an output and ran the Basic version shown below. If some of the code looks strange, it is required to create twos complement variables in Basic.

```
10 DIM R(255)
20 DEFFNA(X)=INT(65535*(X/65535-INT(X/65535))+.5)+1
30 DEFFNB(X)=INT(256*(X/256-INT(X/256))+.5)
40 PMODE4,1:PCLS:SCREEN1,1
50 SEED=RND(65536):LAST=RND(59)
60 FOR I=0 TO 1 STEP0
70 LAST=LAST*SEED
80 LAST=FNA(LAST)
90 IF LAST>32767THENLAST=ABS(LAST-65536)
100 R=FNB(LAST)
110 R(R)=R(R)+1
120 PSET(R, R(R))
130 NEXT
```



It is now possible to see that in addition to the zero state problem, the algorithm is highly dependent on the original values of last and seed. Unless these are both prime numbers you do not produce a prng as not all numbers in the sequence can be obtained. This routine would need to be rejected for any serious work.

It is not necessary to test pairs or triplets.

Tom's algorithm is close to that of a linear congruential generator. The algorithm should be: RND(j+i) = MOD(a*RND(j) + c, m) . This routine can not get stuck at zero or any other number. The length of the sequence is M and all numbers from 0 - M are produced when A and C are chosen correctly.

The choice of A, C, and M is difficult and the research is on going. The larger the value of M, the better the prng with one caveat. The value of A*M must not cause a number overflow. This means if we write a prng for C and use integers A*M < 32767; for longs A*M < 2,147,483,647; for doubles A*M < 4,294,967,296.

Luckily for us, tables of a,c,m values are available. Reasonable values are given below.

	overflow	m	a	c
RSDOS	2**30	714025	1366	150889
Basic09	"	"	"	"
C				
long	"	"	"	"
float	2**23	11979	430	2531
double	2**54 use values for			
	2**35	217728	84589	45989
Assembly	use 2**35 or find your own values with multiple precision arithmetic			
	table data from "Numerical Recipes in C"			

What follows are some implementations in C. Tom's algorithm is also retested in the C environment to eliminate any funnies that might have been caused by Basic.

My version of an prng is somewhat slow because the Tandy version of C does not have a modulus function that works on doubles. We need to use doubles to approach a prng of reasonable quality. The new ANSI versions of C do have the double precision % function.

If you try them you will find that true prng routines work just as well in C as the built in routines of Basic although slower. For speed, the routines would need to be written in assembly code.

Please note that @ stands for the C back-slash!!!!!!!

```
PROGRAM #1  should either #include "rnd.c" or -l=rnd.r
/* screen creates a vdg graphics screen and does graphics
  with rnd() as source. Graphs range of rnd. */


main ()
{
   /* vdg 256x192 2color black&white */
   static char scr_on[]="@xf@x0@x5";

   int cleanup();
   intercept (cleanup);
   write(1,scr_on,3);
   draw();
}
```

```
cleanup (sig)
{
   /* deactivate graphics screen */
   static char scr_off[]="@x12";
   write(1,scr_off,1);
   exit (0);
}


draw()
{
   int i,x;
   char p[256],pset[3];
   double rnd();
   pffinit();
   /* set point command */
   pset[0]=24;
   for (i=0; i!=256;i++){
      p[i]=0;
   }
   for (;;){
      /* get double precision rnd and convert to integer */
      x=rnd(256.)-1.;
      /* count times rnd appears */
      ++p[x];
      /* set point x=rnd y=count */
      pset[1]= x;
      pset[2]= p[x];
      write(1,pset,3);
   }
}
```

PROGRAM #2 again must incorporate or call rnd

```
/* screen1 creates a vdg graphics screen and does graphics
  with rnd() as source. This version for rnd pairs. */


main ()
{
   /* vdg 256x192 2color black&white */
   static char scr_on[]="@xf@x0@x5";

   int cleanup();
   intercept (cleanup);
   write(1,scr_on,3);
```

```
    draw();
}

cleanup (sig)
{
    /* deactivate graphics screen */
    static char scr_off[]="@x12";
    write(1,scr_off,1);
    exit (sig);
}


draw()
{
    int i,x,y;
    char pset[3];
    double rnd();
    pffinit();
    /* set point command */
    pset[0]=24;
    x=rnd(192.)-1.;
    for (;;){
        y=x;
        pset[2]=y;
        /* get double precision rnd and convert to integer */
        x=rnd(192.)-1.;
        pset[1]=x;
        write(1,pset,3);
    }
}

PROGRAM #3   not mainline module
/* Random number generator; type linear congruential generator
   To be compiled into .r form for a math library
   by Robert Gault */

double atof();
double next=107839.;
#define a 9301.
#define c 49297.
#define m 233280.
double trunc();
pffinit();

double rnd(r)
double r;
```

```
{
   int pos;
   double low, rnum;
   if (r<0){
      next= -r;
   r= -r;
   }
   /* Next few lines will produce the effect of   double % double */
   next=(a*next+c)/m;
   next-=trunc(next);
   /* next line not part of modulo math */
   rnum=next;

   next*=m;
   /* next is now a modulo value of    m */

   if (r>1){
      rnum*=r;

      /* Now truncate the decimal portion without rounding */
      rnum=trunc(rnum);
      /* Offset value to 1-number instead of 0-(num-1) */
      ++rnum;
   }
   return (rnum);
}

/* trunc truncates decimal double to int, double */
double trunc(number)
double number;
{
   double integer;
   char cmem[30];
   sprintf(cmem,"%.0f",number);
   integer =atof(cmem);
   if (integer>number)
      --integer;
   return (integer);
}

PROGRAM #4    not mainline module
/* rough and ready rnd routine by T.Napolitano */

/* quality of output HIGHLY dependent on next two lines. Values should be
both prime numbers for best results. */
int last=31;
```

```
int seed=37;

/* slight changes needed to interface with graphics drivers */
double rnd(big)
double big;
{
   int range;
   range=big+1.;
   last *= seed;
   if(last<0)
      last= -last;
   return(last%range);
}
```

If all this seems overwhelming, you have just scratched the surface of the subject. For those of you who want more, head for the library and "hit the books."

CAVEAT: All OS-9 programs above run on a VDG screen. This was done to accommodate both Coco 1&2 and the Coco3. If you prefer to use a graphics window you will need slight code modification.

```
10 ' SURPRISE CCOG DEMO FOR RANDOM NUMBERS BY ROBERT GAULT
20 ON BRK GOTO340:ON ERR GOTO 340
30 WIDTH32
40 DIM SLOT(20)
50 PMODE4,1:PCLS1:SCREEN1,1
60 CIRCLE(126,20),100,0,.2,.5,1
70 DRAW"BM26,20;C0;M+10,0;M122,5;M+0,-4"
80 DRAW"BM131,5;NM+0,-4;M216,20;M+10,0"
90 LINE(226,20)-(226,191),PRESET:LINE-(26,191),PRESET:LINE-(26,20),PRESET
100 FORI=0TO190STEP10
110 LINE(36+I,191)-(36+I,130),PRESET:NEXT
120 FORI=0TO170STEP10:FORJ=0TO100STEP20
130 IFJ<100THENPRESET(41+I,30+J)
140 PRESET(36+I,20+J)
150 NEXTJ,I
160 COLOR0,1
170 FOR L=0TO1STEP0
180 X=126:FORY=5TO15STEP4
190 GOSUB320
200 NEXTY
210 FORY=15TO125STEP10
220 SOUND253,1:GOSUB320
230 R=5*(2*RND(2)-3):X=X+R:NEXTY
240 X=X-R
250 S=INT(X*.1):SLOT(S)=SLOT(S)+1
260 FORY=128TO190-4*SLOT(S) STEP4
270 GOSUB320
280 NEXT
290 CIRCLE(X,Y),3,0
300 IFSLOT(S)=15 THEN330
310 NEXTL
320 CIRCLE(X,Y),3,0:FORT=1TO20:NEXT:CIRCLE(X,Y),3,1:RETURN
330 EXEC&HADFB:RUN
340 WIDTH80
```

OK

Jim's "CoCo" Corner

BULLETIN BOARD SYSTEM

SUPPORTING THE FULL LINE OF

Tandy Color Computer's

COLOR COMPUTER / COLOR COMPUTER 2

TANDY /// Color Computer 3

On line 24 Hours / 7 Days a Week

300 - 1200 - 2400 Baud 8/N/1

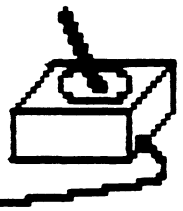
313 - 292 - 4713

Serving you from Taylor, MI.

System Owner/Operator: Jim Snider

*** ON-LINE since December 1987 ! ***

* Now with ON-LINE ordering of CoCoPRO! poducts. *

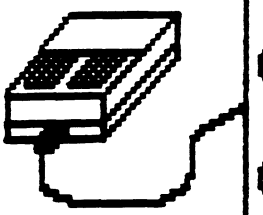


Graphics Utilities Games

PROGRAMING IN: BASIC-OS9 LY2-M/L

Hardware Modifications

WE HAVE SOMETHING FOR EVERYONE



THIS BBS IS SUPPORTED BY THE

Color Computer Owners Group

THE OLDEST COCO CLUB IN THE DETROIT AREA

4-91

C.C.O.G.

COLOR COMPUTER OWNERS GROUP

Oldest COCO Club in Metro Detroit
Meetings the ~~THIRD~~ FOURTH Tuesday of each month
7:00 PM — 9:30 PM
LIVONIA VFW — 7 MILE & MIDDLEBELT
SIDE DOOR — "THE FOXHOLE"

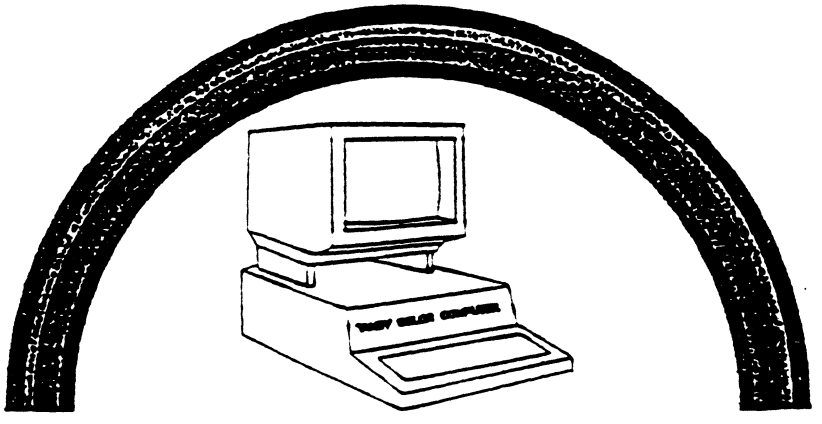
Join other COCO owners for Hardware and Software demos, Public domain Library, Discounts, and general knowledge exchange.

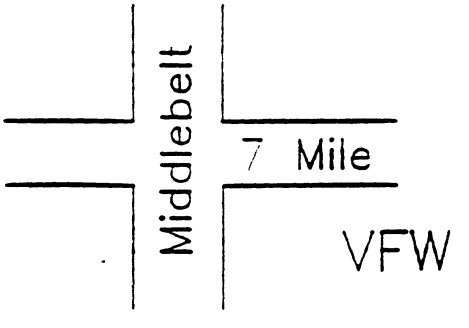
1991 Meetings

Jan 15	***Jul ~~16~~*** 23
Feb 19	***Aug ~~20~~*** 27
Mar 19	***Sep ~~17~~*** 24
Apr 16	***Oct ~~15~~*** 22
May ~~21~~ 28	***Nov ~~19~~*** 26
Jun ~~18~~ 25	***Dec ~~17~~*** ?

MEETINGS INFORMAL AND OPEN TO THE PUBLIC

Contact: Bernie @ (313) 283-2474
or
BBS @ (313) 292-4713





COLOR COMPUTER OWNERS GROUP
C/O BERNIE PATTON
388 EMMONS BLVD.
WYANDOTTE, MI 48192