The Amateur Computerist

Webpage: http://www.ais.org/~jrh/acn/

Fall 1991 **Volume 4 No. 1**

Table of Contents

Computers for the People - A History
or
How Hackers Gave Birth to the Personal Computer
Part II

by Ronda Hauben

(Editor's Note: On August 12, 1991, newspapers and magazines carried articles honoring IBM managers or Bill Gates for having introduced the IBM Personal Computer 10 years ago. But it is not these few "famous men" who are responsible for the breakthroughs in computers that represent the personal computer revolution. The serialized history begun in the last issue [vol.3 no.4] describes how computer hackers and

hobbyists forced IBM to develop and introduce the personal computer in 1981. The second part of this 4 part history follows as our effort to honor the 10th anniversary of the victory of these hackers and hobbyists and their continuing efforts to develop the personal computer and make it available to people.)

The history of the invention of the personal computer is a long one – beginning in the 1600's when the first known mechanical adding machine was built. But technology then was not at the point where a feasible design could be developed and produced. It was not till WWII that technological development had given birth to the components that would make a working computer possible. As part of the war effort, the American government was willing to provide the large sums of money needed to develop and build the ENIAC, the first working computer. ENIAC was a huge monster of a machine. Stan Augarten, in his book ***Bit by Bit***, (New York, 1984, p. 128) tracing the history of computers describes the ENIAC: "It took 2 days to set ENIAC up to carry out a program. You did not sit down at a computer terminal and type in instructions, instead you set thousands of switches and plugged in hundreds of cables [like the cables on an old telephone switchboard -ed] by hand one at a time and wiring a complicated program could take months."

The ENIAC had been developed in 1945. During the 1940's and 1950's there were significant advances in the capability of the computer. Yet into the 1950's, computers were still the exclusive domain of a few. Until the 1960's, as Augarten points out, "most of the computers used were built in the ENIAC'S giant mold. Only the largest institutions – universities, corporations, research institutes and government agencies – could afford them.... Big and costly, they were the very symbols of entrenched and centralized power – haughty, impersonal, inefficient and inaccessible." (p. 253)

Why were the computers of the early postwar WWII period inaccessible to the average person? John Kemeny, one of the inventors of the programming language BASIC describes the dilemma:

"For the first two decades of the existence of the high speed computer, machines were so scarce and so expensive that man approached the

computer the way an ancient Greek approached an oracle.... A man submitted his request – then waited patiently until it was convenient for the machine to work out the problem. Only specially selected acolytes were allowed to have direct communications with the computer. In the original mode of using computers, known as batch processing, hundreds of computer requests were collected by the staff of a computer center and then fed to the machine in a batch."
(quoted in Augarten, from Kemeny's book *Man and the Computer*, N.Y., 1972, p. 21)

Digital Equipment Corporation (DEC) introduced the first mini-computer in 1963 – called the PDP-8. It cost only a fraction of the price of a mainframe computer, but it still cost $18,000. And it ran only one program at a time. (See Augarten, p. 254-255, and 257.)

By 1971, the market for the mini-computer in businesses and universities had taken off. In that year Intel, a semi-conductor company was working on a chip to run a calculator. In the process they developed the 8008 chip, which was powerful enough to run a small computer.

By the early 1970's, Integrated Circuit (IC) technology had reached the point where IC's were sophisticated and inexpensive enough to make a personal computer possible. Many computer companies, DEC and the other large mini-computer makers, had the resources available to develop the personal computer. Technically the task was not complicated, as the logic chips or microprocessors had been developed to the necessary stage. But the big computer companies were not interested.

In the early 1970's, David Ahl, an engineer, worked for DEC in its marketing research department. The company had one half of the educational computer market in 1973 and had $20 million in sales. Ahl moved into a research and development group. After some investigation, he suggested to his superiors that there was a substantial market for small computers in schools and homes. On May 17, 1974, Ahl, presented a plan to DEC to develop and market small computers. Ken Olsen, President of DEC, vetoed the plan saying, "I can't see any reason that anybody would want a computer of his own." (John J. Anderson, "David Tells Ahl-The History of Creative Computing", *Creative Computing*, Nov. 1984, p. 72).

Chris Rutkowski, writing in the 10 year anniversary issue of

Creative Computing explains that Olsen's response was typical of corporate reaction to personal computers. He writes:

"Computers were invented as servants of the biggest organizations ever conceived; the superpowers and the multinational corporations. These groups, not exactly known for their accurate vision of the future, supported and embraced the computer because of its prowess at one thing, number crunching and between censuses and actuarial tables, these groups had lots of numbers to crunch.
(Ibid,"The Computer as a Creative Tool", p. 97)

Rutkowski goes on, "So specialized were the capabilities and so exorbitant the price of these earliest computers that some nameless sage at UNIVAC predicted that the total world market for computers was five." The writer observes, "How this number was arrived at remains unclear today, but the magnitude of his error was soon clear." (Ibid.)

Nancy Stern, in her book *From ENIAC TO UNIVAC* also discusses the reasons large U.S. companies failed to develop computer technology. She suggests that it was not just the narrowness of their vision of the future. She writes:

"Such major industrial firms as NCR, RCA, and IBM were engaged in small scale research on the feasibility of electronic counters and electronic digital computers in the early 1940s. During the late 1940s, these organizations demonstrated an interest in the work of Eckert and Mauchly [developers of the first post WWII computers like ENIAC and UNIVAC - ed] and in other groups designing computers. Such firms did not, however, undertake major computer projects for commercial purposes during the immediate postwar period."
(*From ENIAC to UNIVAC*, Bedford, Ma., 1981, p173)

She goes on to quote from an article by Henry Tropp "The Effervescent Years" (*IEEE Spectrum*, Feb. 1974, p. 76) where Tropp shows how big corporations refused to develop high technology. Tropp describes what happened:

"The National Cash Register Company offers a particularly intriguing industrial 'might have been.' NCR actually had an electronic computing device constructed during the late 1930s. It was a high-speed arithmetic machine which could add, subtract and multiply electronically, and presumably this machine could have become the first

commercial electronic computer had the company wished to pioneer in this field. However, NCR management was not interested in automatic computing per se, but only in improving its existing line of office equipment."
(quoted in Stern, Ibid.)

Tropp cites other examples of major industrial firms that failed to take any initiative in developing electronic devices:

"If one examines the environment at major U.S. corporations during the 1940s, there is a striking lack of interest in any aspect of automatic computing. I have yet to see a publication from that period that reveals an interest in putting research and development funds into finding better ways of doing automatic computation. At Bell Laboratories, for instance, Stibitz's Model I Calculator might well have been the last in the series if it hadn't been for the pressure of wartime needs and the accompanying government support. IBM was similarly uninterested in advancing the computer art.... The company seems to have been primarily interested in improving existing products and building a strong patent position for possible future applications. I have been able to discern no evidence of a vision of IBM – or any other corporation -- in those days that was comparable to that held by Aiken, Maunchly, Stibitz, Atanasoff, J. Presper Eckert or Wallace Eckert" [pioneers in the 1940s in developing computers - ed].

Nancy Stern observes that "IBM's failure to undertake any research projects in this area is particularly surprising, since it was the major manufacturer of calculating equipment and was the company which would be most affected by the availability of commercial computers." She attributes this anti-high technology policy to the highest levels of IBM management, the Chairman of IBM, Tom Watson, Sr. She explains, "Yet no work on electronic digital computers was undertaken by that organization in the 1940s despite interoffice memos written by IBM engineers recommending such work." Two of the inventors of the ENIAC AND UNIVAC computers, Eckert and Mauchly desperately needed money to continue their pioneering work. They went to IBM and asked for financial assistance. Scientists at IBM, Stern notes, were impressed by the men and their work and recommended the support. "But," she reports, "word came uptown from galactic headquarters to

brush them off. There was [to be -ed], in the words of Watson's decision relayed to the Laboratory, no reasonable interaction between Eckert-Mauchly, and IBM." (Ibid.)

Gordon Bell, who worked at DEC, and helped develop the first mini-computer, the PDP-8, and the first commercial time-sharing system, drew a similar conclusion. The industry, he decided, "was unconcerned about the `science' of computing."

Meanwhile, in 1963, a decision was made at Dartmouth College to make an introduction to computers a part of the Liberal Arts curriculum. Two Dartmouth professors, John Kemeny and Thomas Kurtz, devised what is known as the Dartmouth Time-Sharing System (DTSS) so students could get access to a computer. And they designed the computer language BASIC so that novices could learn to program. Kemeny recalling those years, writes: "We designed a few simple instructions for the lay user to enable him to write his first few computer programs with very little training." (*Man and the Computer*, p. 21)

He goes on to explain the rationale used to develop the programming language BASIC (Beginner's All-Purpose Symbolic Instructional Code.)

"The availability of a language as simple as BASIC has made the learning task so simple that computers have come within the power of every intelligent human being, and time-sharing has made it possible to have direct communication between man and machine." (p. 32)

Thanks to Kemeny and Kurtz, a generation of college students grew up in the 1960's who were exposed to computers through BASIC and time sharing and who thus had a realistic assessment of the potential of the computer and of its limitations.

Ted Nelson was one of these converts. In 1974 he published a book *Computer Lib*. He had to publish it himself. In his introduction to Computer Lib, he wrote: "This book is a measure of desperation, so serious and abysmal is the public sense of confusion and ignorance. Anything with buttons or lights can be palmed off on the layman as a computer. There are so many different things, and their differences are so important; yet to the lay public they are lumped together as 'computer stuff,' indistinct and beyond understanding or criticism. It's as if people couldn't tell apart camera from exposure meter or tripod, or car from

truck or tollbooth. This book is therefore devoted to the premise that EVERYBODY SHOULD UNDERSTAND COMPUTERS. It is intended to fill a crying need. Lots of everyday people have asked me where they can learn about computers, and I have had to say nowhere."

Nelson describes the mystification of computer knowledge. "Knowledge is power and so it tends to be hoarded," he writes, "Experts in any field rarely want people to understand what they do, and generally enjoy putting people down. Thus if we say that the use of computers is dominated by a priesthood, people who spatter you with unintelligible answers and seem unwilling to give you straight ones, it is not that they are different in this respect from any other profession. Doctors, lawyers and construction engineers are the same way." (Ibid.) Nelson maintains this stranglehold on computer knowledge had to be broken. He writes:

"But computers are very special, and we have to deal with them everywhere, and this effectively gives the computer priesthood a stranglehold on the operation of all large organizations, of government bureaux, and anything else that they run. Members of Congress are now complaining about control of information by the computer people, that they cannot get the information even though it's on computers. Next to this it seems a small matter that in ordinary companies `untrained' personnel can't get straight questions answered by computer people; but it's the same phenomenon." (ibid.)

He makes his plea:

"It is imperative for many reasons that the appalling gap between public and computer insider be closed. As the saying goes, war is too important to be left to the generals. Guardianship of the computer can no longer be left to a priesthood. I see this as just one example of the creeping evil of Professionalism, the control of aspects of society by cliques of insiders. There may be some chance, though that Professionalism can be turned around. Doctors, for example, are being told that they no longer own people's bodies. And this book may suggest to some computer professionals that their position should not be as sacrosanct as they have thought, either." (ibid.)

The refusal of major corporations to develop personal computers, meant the task fell to those with a democratic vision – to computer hobbyists and hackers. For example, Jonathan Titus was a hobbyist who

had developed a keen interest in electronics and in tinkering and had a sense of the importance of the microprocessor. (There were previous attempts to create a home computer by folks like Nat Watsworth with his Scelbi-8H, or Nelson Winkless, with his Digi-Comp I. See Stephen Gray, "The Early Days", *Creative Computing*, Nov. 1984, p. 12)

When Intel introduced the 8-bit 8008 chip, Titus studied it and realized that it was powerful enough to run a small computer. (See Augarten, p. 269). He ordered the 8008 from Intel. The chip cost him $120. With it he received a free applications manual complete with circuit diagrams. He wanted to share his design with other hobbyists. He decided to write a letter to two well known hobbyist magazines *Popular Electronics* and *Radio Electronics* asking if they were interested in running an article on Mark-8, his homemade computer. *Popular Electronics* wasn't interested, but Larry Stickler, the Editor of *Radio Electronics* was excited by the proposal and flew out to Blacksburg, Va. to see Titus's computer. (See Augarten, *Bit by Bit*, p. 269)

"The machine was about the size of a large breadbox," writes Augarten, "...programs had to be entered one bit at a time by flipping a set of toggle switches on the face of the machine." And programs were lost forever when the machine was shut off. But the machine worked. (Augarten, p. 269)

The article announcing the Mark-8 computer by Titus ran in the July, 1974 issue of *Radio-Electronics*. The article was sketchy. If you wanted more information you could send away for a 48-page instruction manual written by Titus and published by *Radio-Electronics* for $5.50. You could also buy the circuit boards for $47.50 from Technique, Inc. a small firm in Englewood, New Jersey. All the other components had to be bought from Intel or other companies. "Altogether, the Mark-8 cost about $250 to build – in addition to a lot of time and trouble," explains Augarten. (p. 270)

But to the 10,000 people who sent for the instruction book or the 2500 who also sent for the circuit boards all the complications were not important.

A significant event had occurred. The grassroots movement to make computers available to the American people had exploded. The world was to be significantly changed.

As Ted Nelson had predicted when he published *Computer Lib*, "I am 'publishing' this book myself, in its first draft form, to test its viability, to see how mad the computer people get, and to see if there is as much hunger to understand computers, among all you Folks Out There as I think." Nelson realized, "The computer field is its own exploding universe."

(to be continued)

Letters to the Editor

Dear R. Hauben:

I'm interested in reading a copy of the *Amateur Computerist*. I saw it mentioned by Andrew Ross in his essay, "Hacking Away at the Counterculture," collected in his (and Constance Penley's) recent anthology, *Technoculture*.

I'll be happy to send a fee or donation for the newsletter, either before or after you send me a copy – just let me know how much to send.

I'm interested in your newsletter because I'm a computer programmer committed to sharing my knowledge with my users, in the belief that such knowledge can lead to a general empowerment. So I'm always interested in hearing about instances of similar activity.

Pam Rosenthal

(Editor's Note: Following is the excerpt from *Technoculture*, By Constance Penley & Andrew Ross, Editors, Cultural Politics, Vol 3, University of Minnesota Press, p. 124.)

"A good example is the crucial role of worker technoliteracy in the struggle of labor against automation and deskilling. When worker education classes in computer programming were discontinued by management at the Ford Rouge plant in Dearborn, Michigan, United Auto Workers members began to publish a newsletter called the

Amateur Computerist to fill the gap. Among the columnist and correspondents in the magazine have been veterans of the Flint sit-down strikes who see a clear historical continuity between the problem of labor organization in the thirties and the problem of automation and deskilling today. Workers' computer literacy is seen as essential not only to the demystification of the computer and the reskilling of workers, but also to labor's capacity to intervene in decisions about new technologies that might result in shorter hours and thus in 'work efficiency' rather than worker efficiency."

To the Editor:

"A Brain for Robots has been created on Amiga Library Disk 411. Join/Stop/Expose the Underground Movement toward a Cybernetic Economy: Republish this message."

Authur T. Murray

TEN COMMANDMENTS OF GOOD NETWORKING

by Novell Network Sysop Mel White

Networks are, simultaneously, one of the biggest blessings for a company and one of its biggest headaches. At their best, they provide inexpensive solutions for a workgroup to share critical files. At their worst, they are cranky fortifications of high tech electronics with mysterious commands and obscure problems that only a technical junkie with a masochistic streak (also known as your Local Area Network Sysop) could love. Here are ten simple rules that will help you get along with your Network and that deity in power, the Sysop. Although these commandments are slanted toward users of Novell Netware, they will apply to most types of networks.

I.– THOU SHALT USE THE NETWORK MENUS UNLESS THOU HATH PERMISSION TO DO OTHERWISE.

It's terribly tempting for someone who has some familiarity with DOS to drop to the operating system and try to do things on a network that one does on a PC. But networks aren't PC's. And although Network DOS is similar to PC DOS, system architecture and memory setup can toss some wrenches in the best-laid plans of the DOS user. Even the simple change directory DOS command done in a lighthearted moment can toss you into an area where the network doesn't recognize any of your commands-- or even your existence. And the only way out of there will be to reboot your workstation.

II.– THOU SHALT NOT PUT SINGLE USER SOFTWARE ON THE SYSTEM.

Single user software may be written in machine code that makes calls to specific memory hooks or hardware addresses. But Netware DOS has some different interrupt calls and its hard drive is structured differently than the standard PC's. Direct calls to specific addresses can cause your network station or the network itself to lock up. In one instance, a single user version of a word processor started writing its code all over the Network's operating system files. Needless to say, Sysop was Not Amused. To be safe, use only network versions of your software.

III.– THOU SHALT NOT USE PC TOOLS OR NORTON UTILITIES OR ANY PC DISK UTILITIES ON THE NETWORK.

The network's drive has a different organization structure and logical architecture. And while these are wonderful packages, they expect to find the File Allocation Tables and other information in specific locations. Unfortunately, these are not the places where Netware is keeping these files (by the way, this is why software with the SUPER-LOCK type of copy protection security system won't install on your network).

IV.– IN BUYING HARDWARE AND SOFTWARE FOR THE NETWORK, REMEMBER THAT THY VENDOR KNOWETH BEST.

The cheapest strategy really is to go with the vendor's recommenda-

tion and have them set up under their warranty. Networks can be squirrely beasts and nothing can be more frustrating than trying to put a third party PC card on a network and trying to make it work. This particularly comes into play in situations where you're trying to connect the network to several other networks or mainframes. Stick with what the support staff knows will work. Time is money in a business environment and it's not cost-effective to bring a network to a complete halt while you work for three weeks trying to get a board manufactured by HomeBilt Hardware to work with your network software.

V.– BE COURTEOUS ABOUT DATA.

These are shared files and shared resources. Be aware of the limitations of your system. It won't let you update John Dillinger's record at the same time another user is trying to update it. If several people need to be working in the file at the same time, check to see that no one is working on the same data.

VI.– THOU SHALT NOT SURPRISE THE SYSOP WITH NEW SOFTWARE.

You may be madly in love with the latest version of WordWhapper, but if it's not supposed to go on the network, then Sysop will not appreciate finding it there. Sysop will erase the files and send two large guys from the Personnel Department to explain to you that Sysop does not appreciate having the network's disk space eaten up by unauthorized software.

VII.– THOU SHALT NOT DISCONNECT THY WORKSTATION FROM THE NETWORK BY THYSELF.

When you share resources through a network, it means that you lose control of a lot of your individual options. Moving your workstation is one of them. This has a tendency to either lock up portions of the network or to crash the network completely; conditions which tend to irritate your local Sysop. Workstations can, of course, can be moved around but it must be done properly and that means involving the Sysop and the folks who did the cabling for the network. Remember, that because of your building's architecture, you may not be able to put the

machine where you would like it. The ambience of your office furniture arrangement may therefore suffer--unless you're a vice president. In that case, they'll rearrange the building itself to please you.

VIII.– LEARN TO USE THY NETWORK UTILITIES.

Sysops will teach you to use the "Kill That Print Job" menu. This is an important one to learn. Sysops get snorky if interrupted in the middle of a technical glitch with a request to kill off a print job.

IX.– THY SYSOP AND THY SUPERVISOR MUST BOTH BE TOLD YOUR PASSWORD.

On occasions, you may be out of the office and the Sysop may have need to test a piece of software from your login id. If the Deities In Charge Of The Network don't have your password, they will change your password (on the Novell system, the password isn't displayed under any of the Supervisor's query menus).

X.– REMEMBER THAT NETWORKING IS SHARING RESOURCES.

Play nicely.

Try This Program

```
10 FOR T = 1 TO 2
20 GOSUB 201
30 NEXT T
40 FOR X = 3 TO 10
50 GOSUB 200
60 NEXT X
70 FOR L = 11 TO 12
80 PRINT "******************" TAB(30);"***"
90 NEXT L
100 FOR H = 13 TO 20
110 GOSUB 200
120 NEXT H
130 FOR B = 21 TO 22
```

```
140 GOSUB 201
150 NEXT B
199 END
200 PRINT " ***"," ***"," ***"
201 PRINT "*****","*****","*****"
210 RETURN
```

(Editor's note for IBM computers: When you are ready to run this, press control printscreen and run this from GWBasic on your printer. Press control printscreen to turn off printing to the printer.)

The USSR and the Computer

Since the 1970's, micro-chips and computers have put on the world's social agenda the promise of a better life. From the simple level like microwave ovens that turn themselves on and off to computer aided medicine and computer controlled robotic manufacturing, the computer age promises fewer needed hours of human labor both on and off the job. This promise is known world wide and desired universally especially by working people.

In the 1980's, especially after their space ships Phobos I and II failed, scientist and engineers in the Soviet Union realized technological innovation and development was hampered by the lack of uncensored communication and by the slow process of getting innovations tried and adopted. Thus, to the pressure of people wanting the computer promised better life, was added the demands of scientists and engineers for more channels of communication and for the multiplicity of ideas and directions. These had so far been stifled by the one party ideology and practice. The result was a greater openness called Glasnost.

The great test of the Glasnost pressured for by the existence of computers, was the attempted intra-governmental coup in the Soviet Union in late August 1991. Realizing that openness was being attacked and thus so was the chance to gain from the computer revolution, the working people and students of Moscow and Leningrad rallied to protect

their gains. Newspapers that the coup leaders attempted to censor, defiantly distributed calls for resistance and the defense of the uncensored exchange of information. Leaflets were printed with personal computers and passed out in subway stations and on the streets. These papers were eagerly taken up and distributed by people on their way to work. Politicians like Boris Yeltsin in Russia, sensing the strength of the people, championed resistance to the coup and it was toppled.

The computer revolution held out enough of a promise that the people of the Soviet Union rose successfully to defend it. The next challenge to the people of Central and Eastern Europe as well as the rest of the world is to realize that computer and cybernetic development requires the types of innovation, hacking and diversity characteristic only of amateurs in close touch with each other. This requires the continuing struggle for uncensored and widely distributed communication. The governments and profiteers of every society want the advantages of the computer revolution for their own purposes. So, in every society the promise of computer improvements will remain a dream for workers unless there continues to be the fight for openness, support for innovation and the challenging of previous authority. This makes the defense of amateur computing and innovating worth the support of all working and progressive people.

by Jay Hauben

Command Line Calculator

CCalc.C is a Quick C variation of Michael J Mefford's Compute.Com which appeared in the May 29 1990 issue of *PC Magazine*. I liked the concept but wanted more functions such as the elementary trigs, powers & roots.

This program will operate as a command line calculator by entering the following command: CCalc <expression> <return>

Examples: CCalc 24 + 2
CCalc 24 + (tan50) / 2
CCalc -24 + ((TAN(50)) / 2)
CCalc 2 + -3
CCalc 2 + (-3)

As in Mr Mefford's original, CCalc makes available the results of the last calculation in the form of the variable "x". Unlike the original, the program references a small (8 byte) file called ccalc.dat which will be created in the home directory of ccalc.exe if it does not exist. To use this variable, substitute it where needed in the command line, such as: ccalc 24 + xor ccalc 24 + (X)

Spaces, upper/lower case and added parentheses cause no harm, but the parents should be balanced, and, of course, follow the accepted rules of mathematical precedence.

Basically, the program parses a command string assembled from the user input. Spaces in input cause no harm as the individual arguments are concatenated into one string. This string is used by Rewrite() as the source a modified version called achWorkstr. The changes made here include changing all uppercase to lowercase, replacing any trig references such as Sin by a hi-ASCII substitute (char 224 thru 229), replacing any {} or [] by (), checking parentheses balance, and substituting a placeholder for any references to the variable X.

Getparents() is recursively called to identify and send to Parse() the substring consisting of the innermost, last set of parents for subsequent evaluation by eval_string(). After each string is evaluated, it's value is stored in adReal[] and a placeholder consisting of an Uppercase Ascii char is placed in the string. After all parentheses are evaluated in this fashion, the complete string is sent to Parse() for one final go-round. Any time during the evaluation process that an uppercase (A..Z) character is encountered, the corresponding value is used from the adReal array in calculations.

Error checking is modest, limited to divide by zero and parent balance but the program is relatively robust and does not hang even when sent some nonsense like CCALC 3+-*/sin20. Also, I have attempted to use a variation of the "Hungarian Notation" as presented by Charles Petzold in his column in the March 14 1989 edition of *PCMag*.

I find it useful when the pointers and arrays begin to overwhelm me. The program name can be changed (at least in MS-DOS 3.x) and the "dat" file will change to suit and stay in the directory where the "exe" lives.

Comments, suggestions appreciated to Larry Ritzert, CIS 72317, 1061

```
/* CCalc.c 1.01 Command-line calculator like PC Magazine's
"Compute.Com". Larry Ritzert, 28686 Swan Island, Grosse
Ile, MI, 48138, May 31 1990
    This Quick C variation offers several additional math
functions such as common trigs, powers & roots & float
mod. Also creates (if necessary) a tiny data file in the
directory where CC.exe is located for storage & retrieval
of previous result. */

iPerformed = 1;  }
int is_math_operator(char ch)#include<stdio.h>
#include<string.h>
#include<math.h>
#include<ctype.h>
#include<stdlib.h>

#define PI 3.14159265359
#define deg(x) 180*x/PI
#define rad(x) PI*x/180
#define MAXSTR 256

char *pchInstr, achWorkstr[MAXSTR];
char *pchStr_Begin, *pchStr_End;
 /* global begin & end of string */
double adReal[26] = {0.0};
 /* to store intermediate calcs */
double dValue, *pdVal;
int iR_Ctr = 0, iWS_Len, iPerformed;
FILE *fp;

main( int argc, char *argv[])
{  int iCtr;
  char achPathf[_MAX_PATH],
  achDrive[_MAX_DRIVE],
  achDir[_MAX_DIR], achFName[_MAX_FNAME],
  achExt[_MAX_EXT];
```

```
  pdVal = &dValue;

  if (argc < 2)
    {  say_hi();
      help();
      exit(0);  }
  else /* gather the command line string */
    {  say_hi();
      pchInstr =  argv[1];
      for (iCtr = 2; iCtr < argc; iCtr++)
      strcat(pchInstr, argv[iCtr]);  }
   iCtr = 0;
   strcpy( achPathf, argv[0]);
 /* find "exe" file and make "dat" file */
   _splitpath( achPathf, achDrive, achDir,
   achFName, achExt );
   _makepath( achPathf, achDrive, achDir,
   achFName, "dat" );
   if((fp = fopen( achPathf, "r+b")) ==
   NULL)
   fprintf(stderr, "\n\aCan't read Data
   File, \"x\" invalid");
   else
     if(!(iCtr = fread(pdVal,sizeof(double),
     1, fp)))
     fprintf(stderr, "\n\aBad Read -Data
     File, \"x\" invalid");

   rewrite_string();
   getparents();
   pchStr_Begin = achWorkstr;
   pchStr_End = pchStr_Begin + iWS_Len;
   parse(pchStr_Begin,pchStr_End);
   printf("\n%s = %7.7f",pchInstr,
   adReal[iR_Ctr-1]);
   *pdVal = adReal[iR_Ctr - 1];
   rewind( fp );
   if (!(iCtr = fwrite(pdVal,
   sizeof(double),1,fp)))
   fprintf(stderr, "\n\aERROR - No write
   of \"x\" to data file");
   fcloseall();  }

rewrite_string()
```

```
 /* cleans up orig input, spaces it *
   * rewrite curly, square brackets */
{  char *pchLocalWS;
   char *apchTrig[6] =
   {"tan","sin","cos","atn","asn","acs"};
   int  iCtr, iLefts = 0, iRights = 0;

   strcpy(achWorkstr, pchInstr);
   strlwr(achWorkstr);
   for (iCtr = 0; iCtr < 6;iCtr++)
  /* replace "sin", etc w/ hi-ascii char */
      {  while(pchLocalWS = strstr(achWorkstr,
      apchTrig[iCtr]))
         {  memset(pchLocalWS, 224 + iCtr, 1);
            /* ch 224 = tan */
         memmove(pchLocalWS+1, pchLocalWS+3,
         strlen(achWorkstr) - 2); } }
   pchLocalWS = achWorkstr;
   while (*pchLocalWS != '\0')
 /* dump squares & curlies */
      {  if (*pchLocalWS == '{' || *pchLocalWS ==
    '[' || *pchLocalWS == '(')
         {  memset(pchLocalWS,'(',1);
         iLefts++;  }
      if (*pchLocalWS == '}' || *pchLocalWS
      == ']' || *pchLocalWS == ')')
         {  memset(pchLocalWS,')',1);
         iRights++;  }
      pchLocalWS++; }
   if (iLefts != iRights)
 /* check for parent balance */
      {  fprintf(stderr,"\n\a%s   Unbalanced
      Parentheses", pchInstr);
      exit(0);  }
   while (pchLocalWS = strchr( achWorkstr, 'x'))
      /* using prev answer? */
      {  adReal[iR_Ctr] = dValue;
       /* assign value to float array member...*/
       memset(pchLocalWS,iR_Ctr + 'A',1);
        /* create a placeholder in string */
       iR_Ctr++;  }  }

getparents()
   /*-----recursively called to evaluate each set of
```

```
     parents---*
    *--starting with the last, innermost set----*/
{  char *pchLocal;
   iWS_Len = strlen(achWorkstr);
   if((pchLocal = strrchr(achWorkstr,'('))
   == NULL)
      return(0);
      /* there are no (longer) any parentheses */
   else
      pchStr_Begin = pchLocal;
      pchStr_End = strchr(achWorkstr +
      (pchStr_Begin-achWorkstr), ')');
   *pchStr_Begin = ' ';
    /* dump this set of parents */
   *pchStr_End = ' ';
   parse(pchStr_Begin,pchStr_End);
    /* send substring prev in parents */
   getparents();
    /* keep checking for parents */  }

parse(char *pchStr_Begin,char *pchStr_End)
      /*  check each (sub)string in  *
       *  math precedence order & from left *
       *  to right for math operators  *
       *  store intermediate results in  *
       *  "real" array */
{  char ch, *pchOper_Pos;
   int  iIndx, iStrLn = pchStr_End -
   pchStr_Begin, iTest;
   iPerformed = 0;
   iTest = 0;
   for(iIndx = 224; iIndx < 230; iIndx++)
      while((pchOper_Pos =
      memchr(pchStr_Begin, iIndx, iStrLn)))
         eval_string( pchOper_Pos );
   while ((pchOper_Pos =
   memchr(pchStr_Begin, '^', iStrLn)))
      eval_string( pchOper_Pos );
   for (iIndx = 0; iIndx <= iStrLn; iIndx++)
      if (((ch = *(pchStr_Begin + iIndx)) ==
      '*') || (ch == '/'))
         {   pchOper_Pos = pchStr_Begin + iIndx;
         eval_string( pchOper_Pos );
         iIndx = 0;  }
```

```
   while ((pchOper_Pos =
   memchr(pchStr_Begin, '%', iStrLn)))
      eval_string( pchOper_Pos );
   for (iIndx = 0; iIndx <= iStrLn; iIndx++)
      if (((ch = *(pchStr_Begin + iIndx)) ==
      '+') || (ch == '-'))
         {   pchOper_Pos = pchStr_Begin + iIndx;
         eval_string( pchOper_Pos );
         iIndx = 0;  }
   if (!(iPerformed))
    /* no operator or function in string */
      {  for(iIndx = 0; iIndx < 26;iIndx++)
         if (strchr(pchStr_Begin,65 +
         iIndx))
            {  iTest = 1;
             /* don't convert lonesome placeholder */
            break;  }
      if (!(iTest))
         {   adReal[iR_Ctr] =
         atof(pchStr_Begin);
         memset(pchStr_Begin, iR_Ctr + 65,
         1);
         iR_Ctr++;
         memset(pchStr_Begin + 1, ' ',
         iStrLn - 1);  }  }
   return(0);  }

eval_string( char *pchOper_Pos )
{  int iTest, iCh, bUnary;
   unsigned char uchOperator;
   char *pchLeft, *pchRight;
   double dFirst, dSecond;

   pchLeft = pchOper_Pos;
     /* set left pointer and... */
     /* backtrack until something found */
   uchOperator = *pchOper_Pos;
   if((uchOperator == '+') || (uchOperator
   == '-')
      && ((*(pchOper_Pos + 1) == '+') ||
      (*(pchOper_Pos + 1) == '-')))
      bUnary = 1; /* two +- in row, assume
            unary */
```

```
   while (!(is_math_operator(*(pchLeft-1)))
   && (pchLeft > pchStr_Begin))
      pchLeft = pchLeft - 1;

   while (isspace(*pchLeft))
    /* in case over shot into space */
      pchLeft = pchLeft + 1;

   iCh = (int)*pchLeft;

   if (pchLeft == pchOper_Pos)
      dFirst = 0.0;
       /* got to start, must be unary value */
   else
      if (isupper(*pchLeft))
       /* is upper, must be placeholder */
         dFirst = adReal[iCh - 65];
          /* use it's stored value */
      else
         dFirst = atof(pchLeft);
          /* else convert the substring */

   pchRight = pchOper_Pos + 1;
    /* now to the right */

   iCh = *(pchRight);
   while ((isspace(iCh)) && (pchRight <
   pchStr_End))
      iCh = *(pchRight + 1);

   if (isupper(iCh))
    /* now for second operand */
      dSecond = adReal[iCh - 65];
   else
      dSecond = atof(pchOper_Pos + 1);
   if (bUnary)
      pchRight = pchRight + 1;
   while (!(is_math_operator(*pchRight)) &&
   !(isspace(*pchRight))
         && (pchRight < pchStr_End))
      pchRight = pchRight + 1;
       /* find right end of operand */

   memset(pchLeft, iR_Ctr + 65, 1);
```

```
    /* create a placeholder */
   iTest = pchRight - pchLeft;
    /* ...insert it in the string */
   memset(pchLeft + 1, ' ', iTest-1);
              /* overwrite w/ spaces */
   store_value(uchOperator, dFirst,
   dSecond);  }

store_value(unsigned char uchOper, double
 dFirst, double dSecond)
{  /* store a float value for each placeholder created in
the string */
   int iTest;
   switch (uchOper)
    {  case '+': adReal[iR_Ctr++] = dFirst
      + dSecond; break;
      case '-':  adReal[iR_Ctr++] = dFirst
      - dSecond; break;
      case '*':  adReal[iR_Ctr++] = dFirst
      * dSecond; break;
      case '^':  adReal[iR_Ctr++] =
      pow(dFirst,dSecond);  break;
      case '%':  adReal[iR_Ctr++] =
      fmod(dFirst,dSecond); break;
      case 224: adReal[iR_Ctr++] =
      tan(rad(dSecond)); break;
      case 225: adReal[iR_Ctr++]
      =sin(rad(dSecond)); break;
      case 226: adReal[iR_Ctr++] =
      cos(rad(dSecond)); break;
      case 227: adReal[iR_Ctr++] =
      deg(atan(dSecond)); break;
      case 228: adReal[iR_Ctr++] =
      deg(asin(dSecond)); break;
      case 229: adReal[iR_Ctr++] =
      deg(acos(dSecond)); break;
      case '/':   if (dSecond == 0) {
         printf("\n\a%s Divide by Zero
         Error",pchInstr);
         exit(0); }
      else
         adReal[iR_Ctr++] = dFirst /
         dSecond; break;
      default :   {
```

```
         printf("\n\aUndefined Error");
         exit(0); }  }

i /* Pascal's "set" would be nice */
{  int iCtr;
   char legals[] = {'+','-','*','/','^','%','α','ß','Γ',
   'π','Σ','σ'};

   for(iCtr = 0; iCtr < 12; iCtr++)
      if(ch == legals[iCtr])
         return(2);
   return(0);  }

say_hi()
{   printf("\nC-Calc 1.01 by L W Ritzert,
   Grosse Ile Mi CIS 72317,1061");  }

help()
{  printf("\nSyntax:  C-CALC <arithmetic
   expression>"
      "\nOperators: x () {} [] + - * / % ^
      tan sin cos atn asn acs 0..9"
      "\n\t'x' = previous result"
      "\n\t'%%' = modulus operator for
      integer or real value"
      "\n\t'^' = power operator, use
      fraction for root"
      "\n\tSpaces, upper/lower case
      optional"
      "\n\tAngles in degrees\n");  }
     /******CHANGES********************
     JUN 3 90 Changed fopen( fp ) to r"r+b"
     to correct bug in file read (binary).
     Also modified fread (fp) verify, both
     chgs n MAIN() */
```

The Question of Censorship

by Michael Hauben and Ronda Hauben

(Editor's Note: This article is available to readers censored or uncensored. If your newsletter contains the censored version, send a Stamped Self Addressed Envelope to R. Hauben, P.O. Box 4344, Dearborn, MI 48126 and the uncensored version will be mailed to you.)

It is hard to believe censorship of the *Amateur Computerist* is now happening when the *Amateur Computerist* was originally started to oppose censorship. The article itself began with a discussion of censorship and government censorship.

Censorship is a tool that reminds me of the past. Censorship is a form that squelches opinions. Of all publications, I would never conceive the *Amateur Computerist* could censor. In our previous issue, vol. 3 no. 4, there was an article titled "Computer BBS Discussion On The War." The members of the *Amateur Computerist* agreed to print it because it was representative of the debate about the war against Iraq on computer BBS's, and it presented many different views and opinions. However, there was one discrepancy. One half of the staff members of the *Amateur Computerist* objected to the included profanity. This led to an internal discussion. The members of *Amateur Computerist* in opposition to any form of censorship tried to have a debate where we presented our side of non-censoring of ideas.

The side that censored the article identified certain words in the discussions that they considered profane and since the *Amateur Computerist* is available for all ages to read, they felt we shouldn't print those "obscene" words. The side opposed to censorship felt that the words weren't being used as obscenities, but rather to convey ideas or to recount what had occurred. To omit these words thus was to censor the ideas or the details of what was being described.

Finally it was agreed that the article would be published, but with the words in question asterisked out. However, a note was included indicating a difference of opinion concerning the censorship and this

article was the result. We at the *Amateur Computerist* would still appreciate our reader's opinions on this subject matter. So feel free to write.

The references in question are as follows:

8. Well, I listened to some of the speakers, and I listened to the a****** standing next to me.

19. They all said they were p***** off and worried about the protests... Their result will just freak the troops out, they think they will be spit at. This is such b*******.

20. ...this is the same b******* that protestors heard in the 60's & 70's.

21. Read the accounts of the troops who returned home to cries of "baby-killing m**********."

22. ...they would not have been called baby-killing m**********.

37. ...and she got in a lot of s*** for it too ....

39. ...she got is s*** from the Hayden/Fonda crowd

Where m********** is used, the people aren't saying it, they are repeating it as part of an idea. Elsewhere people are expressing themselves and for the most part it is small compared to the whole discussion. We didn't highlight the profanity until we censor it with the asterisks. When I first read the article I didn't even notice the profanity.

The opinions expressed in articles are those of their authors and not necessarily the opinions of the *Amateur Computerist* newsletter. We welcome submissions from a spectrum of viewpoints.

ELECTRONIC EDITION

ACN Webpage:

http://www.ais.org/~jrh/acn/

All issues of the *Amateur Computerist* are on-line.
Back issues of the *Amateur Computerist* are available at:
http://www.ais.org/~jrh/acn/Back_Issues/

All issues can be accessed from the Index at:
http://www.ais.org/~jrh/acn/NewIndex.pdf

EDITORIAL STAFF

The *Amateur Computerist* invites submissions.
Articles can be submitted via e-mail: jrh@ais.org