The Icon Newsletter

No. 39 – August 15, 1992

Contents

Feedback

In the last *Newsletter*, we asked persons who wanted to maintain their free subscriptions to return a renewal form. We've received lots of these and they're still coming in. If past experience is repeated, we'll get strays for years.

If you receive this *Newsletter*, you can be sure that your subscription will continue. (Persons who inquire about Icon automatically get a subscription.)

Many persons who responded also filled out the questionnaire on the renewal forms, and some enclosed detailed notes and questions. We tried to answer all specific questions; if you think you should have heard from us, but have not, please send us a reminder.

We have not compiled detailed statistics from the questionnaires, but we read them all and noted several things of interest.

For one thing, the vast majority of subscribers to the *Newsletter* use MS-DOS, with a significant percentage running on 386 or 486 platforms. The other platforms on which Icon is used the most are the Macintosh and UNIX workstations.

Many persons commented that they are new to Icon or classify themselves as novices. We'll try to keep this in mind in the material we include in future *Newsletter*s.

A surprisingly large number of persons said they like the *Newsletter* as it is. This is, of course, gratifying to us and we appreciate all the kind remarks in this regard.

The major topic some persons wanted to see given more emphasis in the *Newsletter* is programming, especially simple examples. We'll try to give more space to this subject in the future.

See "From our Mail", starting on page 8, for specific comments and other noteworthy material taken from the questionnaires.

New Implementations

We're up to Version 8.7 of Icon. We use minor version numbers (8.5, 8.6, 8.7 ...) to keep track of minor changes, so that if there's a problem we know which version of our source code to check.

The difference between minor versions often is not enough to warrant new releases (we went all the way from 8.0 to 8.5 without updating any of our distributions). For example, the main differences between 8.5 and 8.7 are in the implementa-

tion and not enough to warrant a new release for MS-DOS, which came out earlier this year in Version 8.5. (See *Newsletter 38* for information about Version 8.5 and hence 8.7.)

We have, however, updated Icon for several platforms whose last release was for Version 8.0. We hope to have other updates before long. Version 8.7 for VMS is nearly done. For other platforms, we depend on individuals who contribute their time and effort freely and with whom it's sometimes difficult to exchange technical information. Consequently, these implementations may take longer.

The implementations listed below can be ordered as described on pages 13-15 of this *Newsletter*. They also are available via FTP. Except for the very large UNIX system, they also are available from our RBBS. See the article on page 3 and the box on page 7. *Note:* Some of the implementations listed below have been available for some time. If you received Icon from us recently, check to see if it's one of these before re-ordering.

Version 8.7 for UNIX

Version 8.7 of Icon is now available for UNIX. This system includes the optimizing compiler in addition to the interpreter. It also includes graphics support via access to X Window System facilities as described in *Newsletter 36*.

Version 8.7 for OS/2

Version 8.7 is available for OS/2 2.0 as a 32-bit application. Neither the compiler nor graphics support is included.

New MS-DOS 386/486 Release

Although MS-DOS Icon for 386 and 486 platforms is still at Version 8.5, its support for various memory environments has been improved. It now runs with

MS-DOS 5.0's EMM386

MS-DOS 5.0's DOS=HIGH

MS-DOS 5.0's HIMEM.SYS

Helix Software's NETROOM v2.1

Qualitas' 386MAX v6.0

Quarterdeck's QEMM 386 v6.0

The current release supports the VCPI standard but does not run under DESQView 386.

8.0 for the Acorn Archimedes

Paul Moore has provided executable files for Version 8.0 of Icon for the Acorn Archimedes. *Note:* The distribution is on MS-DOS formatted diskettes. Although the capability for reading MS-DOS diskettes is not as yet incorporated in the Archimedes operating system, software to read MS-DOS diskettes on that platform is freely and readily available.

The Icon Optimizing Compiler

As noted in the previous article, the Icon optimizing compiler is now available for UNIX.

In the past, we were uncertain how portable the compiler would be to other platforms. We can now report that the compiler is quite portable and even runs under MS-DOS. We anticipate only routine technical difficulties in transporting the compiler to other platforms.

The compiler does, however, need a lot of resources and specifically a large amount of memory. Under standard MS-DOS, the compiler can only handle programs of trivial size, and so, as we expected earlier, it's not useful on that platform — or on other platforms with limited amounts of memory. However, MS-DOS on 386 and 486 platforms with sufficient extended memory works well, although for acceptable performance, a fast CPU is needed.

One problem with distributing the compiler is that it generates C code, so a C compiler is needed also. We plan to package executable files and libraries for several 32-bit C compilers in a single distribution.

We also plan to include the Icon compiler in the next source update for Icon, so that subscribers to that service can experiment with it.

Changes in Icon Distribution

The variety of magnetic media and the formats in which they are written is large and sometimes bewildering. Things keep changing as the technology of storage media improves. At present, we write 3.5" diskettes at 400K, at 720/800K, and at 1.44M. Some platforms support even higher densities, yet some older platforms can only read the

lowest density. No end to media change is in sight; rather the opposite.

To the extent that we have the facilities, we try to provide Icon on the latest media, while maintaining a bridge for those with older platforms.

Media are not the only problem. The new implementation of Icon for UNIX is an example. Some persons running UNIX on personal computers and workstations can only read diskettes. Depending on the platform, the diskettes could be in one of several UNIX formats or in MS-DOS format, which most UNIX personal computer and workstation platforms can read. There's also the question of 5.25" or 3.5" drives, each with media in two densities. You can see the problem.

We have other kinds of problems. Stocking many different forms of a release is expensive and complicated. It takes much more preparation and it also costs more to ship several low-density 5.25" diskettes instead of, say, one 3.5" high-density diskette.

For a while, we tried to "tough this out". But the situation has gotten out of hand, and we've made some changes to control costs (and to try to maintain our collective sanity). Specifically, we no longer distribute as many different formats as before. For example, UNIX Icon on diskettes is now available only on 3.5" high-density ones. We've also dropped cpio format for UNIX tapes and cartridges, and we now write tapes only at 1600 bpi.

We've also dropped the separate source distribution for porting to new platforms.

We realize that the unavailability of some forms of distribution may inconvenience some users. In most cases, the worst it means is the use of an alternate utility. In some cases, users may have to convert media on another platform. However, if we've made it impossible for you to install Icon on your platform, let us know and we'll try to help.

Icon Via FTP

Recently there's been a substantial increase in the number of files being downloaded from the Icon FTP area. This may be partly a result of increased interest in Icon, but it also reflects the fact that more persons are gaining FTP access.

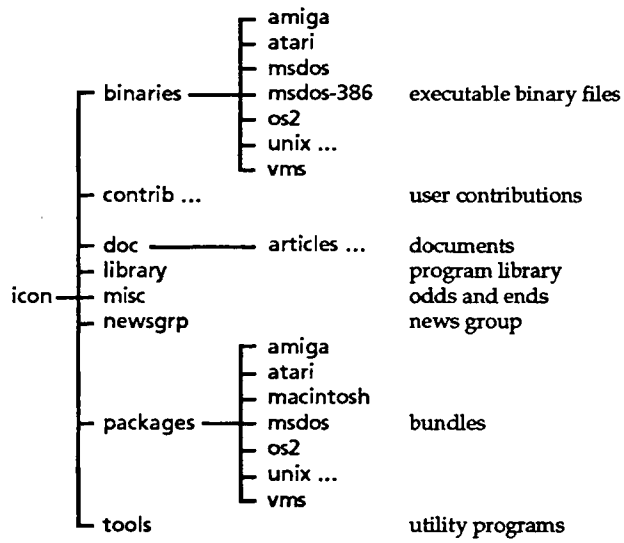
Experienced FTP users seem to have no trouble finding what they want. Persons new to FTP, however, sometimes have trouble getting connected and, when they do, they often have difficulty navigating through the directory structure.

To get started, FTP to cs.arizona.edu. When you are asked to log in, enter anonymous. When you are asked for a password, enter any non-empty string (such as your last name). Then

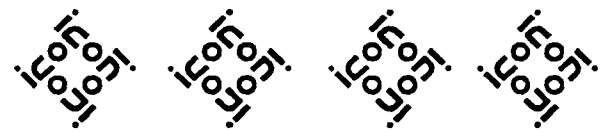
```
cd icon
get READ.ME
```

Look through READ.ME to see what's available. It will tell you what sub-directories there are and what's in them. Sub-directories also have READ.ME files to help in further navigation.

We re-organize our FTP area from time to time as the material we provide changes. Here's the "big picture" of our present FTP area; studying it in advance may save you some time. The ellipses indicate sub-directories not shown here.



The binaries directory contains only executable files. The contrib directory contains various material supplied by users. The doc directory contains documents in a variety of forms. Its articles subdirectory contains selected articles from the *Newsletter*. As you'd guess, the misc directory contains odds and ends we couldn't classify otherwise. Accumulated mail from the Icon news group is in newsgrp. The packages directory contains Icon program material in the same form as it's distributed on magnetic media. The tools directory contains archiving utilities and other useful programs.



An Icon Debugger

Editors' Note: The following article by Charles Shartsis describes an unusual and clever approach to debugging in Icon. Although this article is somewhat technical, we're including it in the Newsletter so that it will reach a large audience.

The high-level facilities of Icon make programming in the language a pleasure. Unfortunately the same cannot be said of debugging an Icon program. Debugging usually consists of inserting calls to write() in and around suspect code. The &trace keyword also can be turned on. However, tracing only displays procedure entrance and exit points. The programmer must still rely on inserting calls to write() and recompiling in order to get really useful debugging information. An interactive source line debugger that allows the display and modification of variables would be a great time saver.

DEBUGIFY

DEBUGIFY is a program, written in Icon, that allows a programmer to produce versions of programs that include interactive debugging capabilities. A DEBUGIFYed program invokes a debugging procedure prior to executing each source code line containing executable code. The debugging procedure is also written in Icon and can be modified to suit individual preferences. Typically the debugging procedure allows the user to enter commands interactively in order to modify or display the state of the executing program. When instructed, the debugging procedure returns control to the DEBUGIFYed program.

Implementation

DEBUGIFY inserts debugging procedure calls directly into a ucode file. When icont is executed with the -c option (compile only; no link phase), intermediate ucode files are produced that can be linked with other ucode files to produce an executable Icon program. Ucode can be thought of as source code written in a relatively simple stack language. For a brief discussion of the structure of ucode files, see "An Imaginary Icon Computer" in Issue 8 of The Icon Analyst (October 1991).

Two aspects of ucode simplify the insertion of debugging hooks. First, ucode is a stack language with a postfix order of evaluation. Roughly speaking, a ucode file consists of a sequence of code segments each of which follows a general pattern. One or more ucode instructions push arguments on an evaluation stack followed by an instruction that performs an action using those arguments. In the process of performing the action, the arguments are removed from the stack. For example, the following code segment assigns the zeroth constant in the constant table as the value of the zeroth variable in the variable table.

```
var     0
int     0
asgn
```

The numbers that identify the variable and integer constant are pushed to the stack by the first two ucode instructions and the asgn instruction performs the assignment while removing the arguments from the stack.

The fact that ucode is based on a stack mechanism makes it easy to insert additional lines into a ucode module without disturbing the original functionality. It is "safe" to insert additional ucode instructions, provided that (1) executing the new instructions leaves the stack in the state it was in prior to executing the inserted code and (2) the values of variables referenced by the original code are not altered by the inserted instructions.

The second aspect of ucode that simplifies the insertion process is that the Icon translator provides source-code line markers in each ucode file. For example, if icont translates

```
x := 4
```

at line 5, the corresponding ucode looks like this:

```
        mark    L1
        pnull
        var     0
        int     0
        line    5
        asgn
        unmark
    lab L1
```

Note that the source line number is preserved in the instruction line 5. For every source-code line *n* containing executable code, the corresponding ucode contains line *n* embedded somewhere prior to the ucode instruction that processes the stack arguments. In this case, the line instruction immediately precedes the final action instruction, asgn. In general, there is no guarantee that the line instruction will immediately precede the action instruction, only that it will occur at some point prior to the action instruction.

DEBUGIFY inserts debugging calls immediately following each line instruction. This has the effect of calling the debugging procedure prior to the execution of most source lines that contain executable code. Where a complete ucode action corresponds to several source lines, the debugging procedure is called only once and prior to the first line. With the exception of certain rare circumstances, this arrangement works out well.

Inserted Ucode

For the debugging procedure to do anything meaningful, there must be more code inserted in a ucode file than simply a procedure call. The types of modifications DEBUGIFY makes will be illustrated using a simple program with identifying line numbers as shown in Figure 1.

```
1   procedure main()
2      local x
3      x := 4
4   end
```

Figure 1. Sample Program

This program and its corresponding ucode is shown in Figure 2. Comments describing the ucode are italicized.

Figure 3 on the next page shows the source code and ucode for line 3 (x := 4) after DEBUGIFY has modified the ucode. The source code equivalent of the ucode inserted by DEBUGIFY also is shown. In the middle of the ucode for line 3, DEBUGIFY inserts the ucode equivalent to the following source code:

```
__vals := []
every put(_vals, variable(!__names))
__debug_proc(&file, <proc_name>,
  &line, __names, __vals)
every __i := 1 to *__names do
  variable(__names[__i]) := __vals[__i]
```

The first two expressions set __vals equal to a list of the values of all the original variables in the program. The third expression calls a debugging procedure called __debug_proc. The name of the debugging procedure inserted by DEBUGIFY is always __debug_proc and cannot be changed without changing DEBUGIFY. The debugging procedure has five arguments: (1) the name of the current source-code file, (2) current procedure

source code/comments	ucode	
1 procedure main()	proc main	
2 local x	local	0,000020,x
constant 4 *end declarations* *file name* *line 1 marker*	con declend filen line	0,002000,1,4 x.icn 1
3 x := 4	mark L1 pnull var int line asgn unmark lab L1	 0 0 3
4 end	pnull line pfail end	 4

Figure 2. Source Code/Ucode Correspondence

name, (3) current line number (the one about to be executed), (4) a list of strings representing the original variable names, and (5) a list of the current values of the original variable names. The last two lists have the same number of elements and elements in each list correspond to each other. That is, __val[3] is the value of the variable whose name is stored as a string in __names[3]. Passing the variables and their values in this fashion gives __debug_proc() the ability to display or modify the value of any variable present in the DEBUGIFYed procedure.

The last expression sets all the variables to the values contained in the __vals list. This expression is necessary to allow __debug_proc() to make modifications to program variables. A procedure call alone cannot modify the values of variables in the calling procedure, since Icon passes arguments by value.

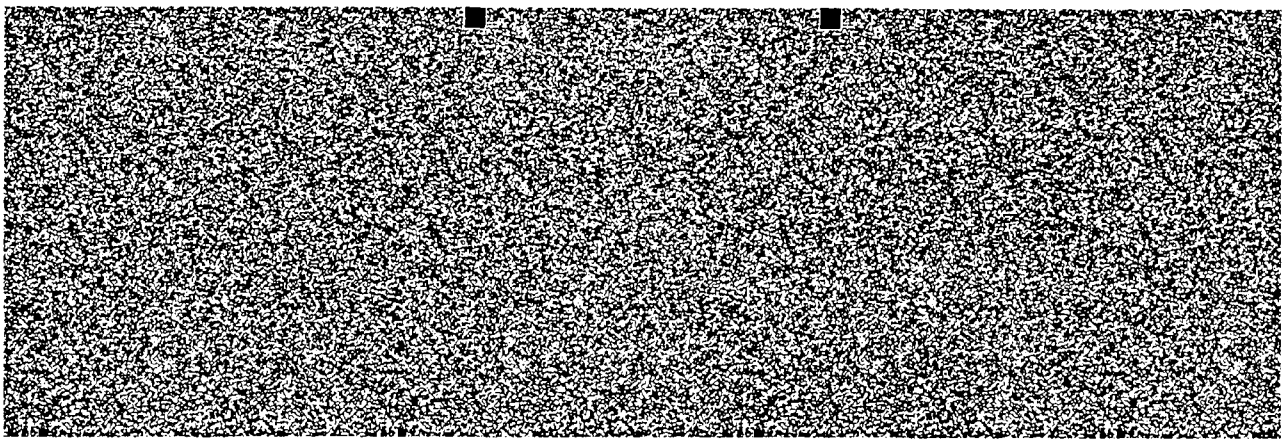
Getting a Copy

DEBUGIFY is available by anonymous FTP from cs.arizona.edu in the directory /icon/contrib. Download either debugify.zip (MS-DOS text line terminators) or debugify.tar.Z (UNIX text line terminators). DEBUGIFY comes with the source code for a ready-to-use debugging procedure.

Editors' Note: DEBUGIFY also is included in the current Icon program library.

3 x := 4 *(first part)*	mark L1 pnull var 0 int 0 line 3
__vals := []	mark L11 pnull var 5 pnull llist 0 asgn unmark lab L11
every put(__vals, variable(!__names))	mark L12 mark0 var 2 var 5 var 3 pnull var 4 bang invoke 1 invoke 2 pop lab L13 efail lab L14 unmark lab L12
__debug_proc(&file, "main", &line, __names, __vals)	mark L15 var 1 keywd 16 str 3 keywd 22 var 4 var 5 invoke 5 unmark lab L15
every __i := 1 to *__names do variable(__names[__i]) := __vals[__i]	mark L16 mark0 pnull var 6 pnull int 2 pnull var 4 size push1 toby asgn pop mark0 pnull var 3 pnull var 4 var 6 subsc invoke 1 pnull var 5 var 6 subsc asgn unmark lab L17 efail lab L18 unmark lab L16
3 x := 4 *(second part)*	asgn unmark lab L1

Figure 3. Ucode Inserted by DEBUGIFY



Icon Auto-Stereogram

The image at the top of this page is an "auto-stereogram" done by Lyle Rains. He sends along these instructions for viewing:

> You will notice two registration dots at the top edge of the image. You need to set the image on a flat, level surface with even lighting. Now look "through" the image as if you were trying to see the floor, and hopefully the registration dots will become "doubled." What you need to do is to make the centermost doubled registration dots merge into a single solid dot, and then focus on the whole image. If/when you see the stereoscopic effect, you will know it. The image will seem like a 3-d relief of raised letters cut out of granite. There will be one solid registration dot in the top center background, flanked by "ghost" dots on either side. Good luck. Not all people seem to be able to see these images, so if you can't see it, try it on someone else.

Editors' Note: If you're not able to see the stereoscopic effect unaided, a stereoscopic viewer, such as is used by engineers and geologists for viewing aerial photo maps, works nicely. Or you may be able to find an old stereopticon in your attic or in an antique store.

Downloading Icon Material

Most implementations of Icon are available for downloading electronically:

BBS: (602) 621-2283

FTP: cs.arizona.edu (cd /icon)

The Icon Newsletter

Madge T. Griswold and Ralph E. Griswold
Editors

The Icon Newsletter is published three times a year, at no cost to subscribers. To subscribe, contact

Icon Project
Department of Computer Science
Gould-Simpson Building
The University of Arizona
Tucson, Arizona 85721
U.S.A.

voice: (602) 621-8448

fax: (602) 621-4246

Electronic mail may be sent to:

icon-project@cs.arizona.edu

or

...uunet!arizona!icon-project

THE UNIVERSITY OF
ARIZONA
TUCSON ARIZONA

and

The Bright Forest Company
Tucson Arizona

From Our Mail

Please use my home address for the Newsletter but my office address for the Analyst.

We realize that many persons use their home and business addresses for different purposes. However, because of the way we manage our address list, it is not practical to maintain more than one address for one person. If you absolutely must have different items sent to different addresses, give us a clear variant of your name for a second address. This is still risky; we can't guarantee to remember why we have two addresses for the same person and one may get deleted when we clean up our mailing lists. By the way, when you place an order for Icon material or otherwise communicate with us and use an address that is different from the one we have on file, we update our data base with the new address unless you specifically ask us not to.

I noticed you have started using LHarc for compressing and archiving MS-DOS Icon files. Is LHarc available for other computers?

We switched to LHarc for MS-DOS material because of its superior compression ability. There are versions of LHarc for other platforms. We have ones for the Amiga, the Atari ST, and UNIX, in addition to MS-DOS. These programs are available from us via FTP and our RBBS.

Please send me a copy of the Icon program for translating PL/I to C that you mentioned in Newsletter 38.

We do not have a copy of this program. It, like many Icon programs we've heard about, is considered to be proprietary by the company in which it was developed. If we had a copy, we would put it in the Icon program library to make it generally available.

Why do you use MS-DOS formatted diskettes for most of your Icon distributions? I can read these diskettes, but it would be more convenient for me to get material in the native format for my Xenix system.

While we realize MS-DOS format is an inconvenience for some persons, using it makes our distribution much simpler in many ways. The use of MS-DOS diskettes is simply expedient for us and allows us to apply more of our resources to other, more important matters.

I'd like to see more information about user applications in the Newsletter.

We're happy to do this — but we can only report about what we know. If you have an interesting application, tell us about it and we'll include it in the Newsletter.

Why don't you publish mini-reviews of freely distributable programs written in Icon? This would help prevent re-inventing the wheel.

Icon programs that are available to us are included in the Icon program library. There are hundreds of files in this library. The best way to find out about programs that might be of interest to you is to get the library and read the documentation there.

I'd like to see more examples of non-exotic code.

We realize we have a tendency to present advanced, clever, and sometimes arcane code. We'll try to remember that many of our readers are relatively new to Icon and would like to see more basic examples.

I'd like to see articles on the details of the implementation of Icon. I realize things like this are covered in the Analyst. I guess I should subscribe.

The *Icon Newsletter* is devoted primarily to topical information about Icon. Many readers of the *Newsletter* are comparative novices as far as Icon goes and would like to see less technical material rather than more. As you noted, material of a more technical nature is included in the Analyst. Technical reports also are available from the Icon Project. IPD117 contains a list of such reports; you can get a copy of it free for the asking.

It isn't necessary to include ordering information for Icon in every issue of the Newsletter. You could replace it with more interesting articles.

See the next item.

Don't drop the order form or the update information from the Newsletter. I like them best.

The two comments above came in renewal forms for the *Newsletter* and arrived in the same mail! We include the order form in every issue because we get many new subscribers who ask for it. Even long-time subscribers sometimes misplace the ordering information and ask us for copies. We don't displace any articles in favor of the ordering information; we publish all the material we have.

I'd like to see more material aimed at SNOBOL4 users who are making the switch to Icon.

This is difficult — we don't have much advice on this subject except to suggest that you learn Icon and not attempt to translate from SNOBOL4.

I'd like to see more material on work in progress in the Newsletter.

We report on work in progress in the *Newsletter* from time to time. The Analyst provides information of a more technical nature on such work.

I'd be willing to pay $10-$20 a year for the Newsletter if that's necessary to keep it going.

We presently think we can continue the *Newsletter* without a subscription charge. Thanks for your willingness to pay if necessary, though.

Please mention in the Newsletter when articles about Icon appear in magazines. (I'm thinking of the article in the January 1992 issue of BYTE.)

Good point; we'll do that. Readers can help with this too — we don't always know about articles related to Icon and appreciate getting copies.

You should publish more code and fewer frequently asked questions.

Well, this issue of the *Newsletter* isn't going to turn you on. See the next item.

I'd like to see more questions from users together with your answers.

"From our Mail" is one of the most popular features of the *Newsletter*. We sometimes include questions that have been asked and answered before, simply because repeated questions indicate the need to repeat the information. And, as we've noted above, every *Newsletter* has many new subscribers who have not seen previous issues.

Can I get the Newsletter on magnetic media?

Speaking of questions that have been asked and answered before ... .This kind of request is made so frequently that we put the current variant of it in "From Our Mail" once or twice a year. There are several reasons we don't publish the *Newsletter* electronically or make it available on magnetic media. Its typographical nature doesn't lend itself to any machine-readable form except PostScript. Such PostScript files are huge. And many persons do not have access to PostScript printers. We have, however, prepared suitable articles from past *Newsletters* as plain ASCII text files and made them available via FTP and our RBBS. See the article on page 3.

I'd like to get better explanations on what's included with the executable files you distribute. For example, what documentation is included?

This has been a long-standing problem for us. The difficulty is the amount of space required for the description. At the risk of offending those persons who don't want ordering information in the *Newsletter* at all, we've expanded the description that goes along with the order form. See page 13.

I'd like more information about Idol, the object-oriented version of Icon.

Idol is included in the Icon program library. There's also a technical report on Idol. You might want to get a copy of IPD117, mentioned earlier; it lists all available publications about Icon.

Please publish more information about the large applications mentioned in Issue 38 of the Newsletter.

In most cases, we do not have a significant amount of information about the applications we listed. In some cases, as mentioned earlier, the applications are proprietary. In other cases we've been given more information as well as the programs themselves for inclusion in the Icon program library. And in some cases, we're still waiting for information that has been promised.

I'd like to see more humor in the Newsletter.

Have you heard the one about the traveling Icon salesman and the programmer's disk drive?

I'd like to subscribe to the Analyst, but I'm just a poor college student.

We realize that the cost of an Analyst subscription is more than some persons can afford or want to pay. The Analyst was initiated largely as a vehicle for providing technical information that was economically infeasible to publish in a free newsletter. Although the cost of the Analyst is a problem for some persons, it's a bargain compared to most other highly technical newsletters. One possibility for you may be to get your college library to subscribe to the Analyst.

Am I free to modify the source code for Icon?

Yes, it's in the public domain. As a courtesy to us and others, you should identify your changes and clearly attribute the source of the original implementation to the Icon Project.

Sixth International Conference on Symbolic and Logical Computing

October 15-16, 1992

Dakota State University, Madison, South Dakota

Computer programming for all kinds of non-numeric applications will be the focus of presentations at the Sixth International Conference on Symbolic and Logical Computing: ICEBOL6.

The Conference will be held on the campus of Dakota State University in rural Madison, South Dakota, on October 15 and 16 Q often the most pleasant time of year in South Dakota.

Ralph Griswold, co-author of the Icon programming language and SNOBOL4, will be a featured speaker at ICEBOL6. He will discuss visualizing program execution in Icon, and he will present a clinic on programming in Icon.

The Banquet Speaker is Robert B. K. Dewar Q the creator of SPITBOL and Realia COBOL; he is currently working on a GNU/Ada compiler.

Mark Olsen is the Keynote Speaker for the Conference. He is the author of many articles, reviews, and papers on computing and literary study. He is the Technical Review Editor of *Computers and the Humanities* and the Assistant Director of the ARTFL Project at the University of Chicago. He will discuss the theory and method of computer application in socio-cultural history.

Among the Conference topics are parsing, conversion of text to hypertext, parallel algorithms for text processing, machine translation, natural-language query processing, artificial intelligence and backtracking, and analysis of the language of Victor Borge.

Proposals for presentations at ICEBOL6 have been accepted from Australia, Bulgaria, England, Finland, France, India, Japan, The Netherlands, Poland, Switzerland, and throughout the United States.

For registration forms and additional information about ICEBOL6, contact

Eric Johnson
114 Beadle Hall
Dakota State University
Madison, SD 57042 U.S.A.

(605) 256-5270

ERIC@SDNET.BITNET

Graphic Credits

The Icon RRubikUs CubeS on the back cover was done by Lyle Rains using Version 2.12 of the ray-tracing program DKBTrace by David Buck. The image as we received it was in full color, but we think even a gray-scale rendition is impressive.

Icon on CD-ROM

Prime Time Freeware has announced Volume 1, Number 2 of its CD-ROM distribution of UNIX-related source code.

This release, which comes on two ISO-9660 CD-ROM disks, contains 3,000 megabytes of UNIX-related source code, documentation, and related material as compressed tar files. A 50-page booklet introduces and explains the disks.

Version 8.6 of Icon is contained in this release, along with more than 200 packages including ABC, AKCL, Andrew, ANU NEWS, Athena, Bertrand, btool, C News, CLU, CLUE, CLX, CMU Common Lisp, COOL, CRISP, Dirt, DJG++, Epoch, EW, Ezd, Franz, Ftptool, GINA, GNU, GO, HyperNeWs, IMAP, INGRES, InterViews, ISODE, JPEG Tools, Kermit (Tapes A-E), LispView, Mach, Magic, MLX, WRL Modula-2, SRC Modula-3, NCSA Data Analysis Tools, NIHCL, Oaklisp, Pari, PCLU, Pine, PlaNet, Postie Pat, Q, Scheme, Scorpion, Serpent, SR, T, tcsh, TIFF tools, Tk, tmnn, UnixTex, URT, VORT, wframe, WINTERP, X11R5, XView, and YAML.

The price for the two disks and booklet is $60. There is a discount for USENIX members and for quantity purchases. Shipping and handling charges are $5 in the United States and $10 overseas. Residents of California add 7% sales tax. Payment must be in U.S. funds by check payable through a U.S. bank, postal money order, MasterCard, or Visa.

To order the CD-ROMs or get more information, contact:

Prime Time Freeware
415-112 N. Mary Avenue, Suite 50
Sunnyvale, CA 94086
U.S.A.

(408)-738-4832 (voice)
(408)-738-2050 (fax)

ptf@cfcl.com

Icon Class Projects

A course on string and list processing is a regular part of the curriculum of the Department of Computer Science at The University of Arizona. This class, which is available to upper-division undergraduate students and graduate students, teaches Icon along with material on programming techniques and data representation for nonnumerical computation.

This spring was the first time that students had a chance to use X-Icon, the new version of Icon that has capabilities for manipulating graphics and text in windows.

Each student did a project in lieu of part of the usual homework and the final examination. There was a wide range of subjects: two source-code analyzers for C, a calendar/reminder application, a text formatter, a maze puzzle with a three-dimensional view, a word processor, a structured program editor for Icon, a graphics interface builder for X-Icon, a chess-playing program, two object-oriented drawing programs, a gradebook application, a coverage matrix calculator, a concurrency modeler, a molecular sequence analyzer, a symbolic differentiator and integrator, a PostScript formatter, a board game, and a macro processor.

Most of the projects turned out quite well, and a few were excellent. Several of the programs will be included in future releases of the Icon program library.

The use of X-Icon was optional for student projects. Despite the fact that producing a project that uses graphics in a significant way almost certainly required additional work, both in learning X-Icon and in doing the project itself, nine of the 19 students elected to use X-Icon. All of the X-Icon projects came out well, and some were impressive indeed.

To give you an idea of what can be done with X-Icon, we're including screen snapshots taken from three of the best X-Icon projects, along with brief descriptions of what the programs do.

The screen snapshots have been reduced considerably to fit in the *Newsletter*. They've also been adapted to print well in black and white, although one of the projects shown here uses color in an essential way.

Figure 1 is a snapshot from the maze program written by Darren Merrill. At the right is a three-dimensional view, in which corridors in the distance are darker. The optional two-dimensional plan view at the left shows the parts of the maze you've traversed. The little arrow shows where you are and in what direction you're headed. Progress through the maze is done with the "arrow" keys on the keyboard, which let you go forward and backward and change direction.

Figure 2 is a snapshot of an interface builder for X-Icon written by Mary Cameron. The menu at the top shows the interface tools that are available. Instances of these tools can be created, placed, and customized. The interface builder can generate an X-Icon application interface, which then can be used in an X-Icon application. Mary is continuing work on her project this summer in conjunction with the development of an X-Icon tool kit.

Figure 3 is a snapshot from an object-oriented drawing program written by Qiang Zhao. This program allows you to draw line segments, polygons, rectangles, and ellipses; each can be in outline form or filled. Objects can be selected, moved, deleted, and their attributes can be changed. Attributes consist of color and texture (stipple).

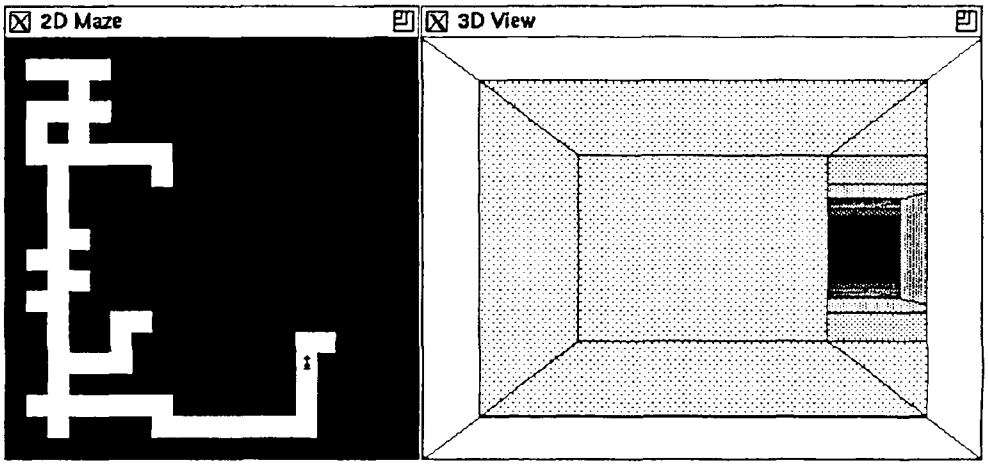


Figure 1 — Three-Dimensional Maze Game

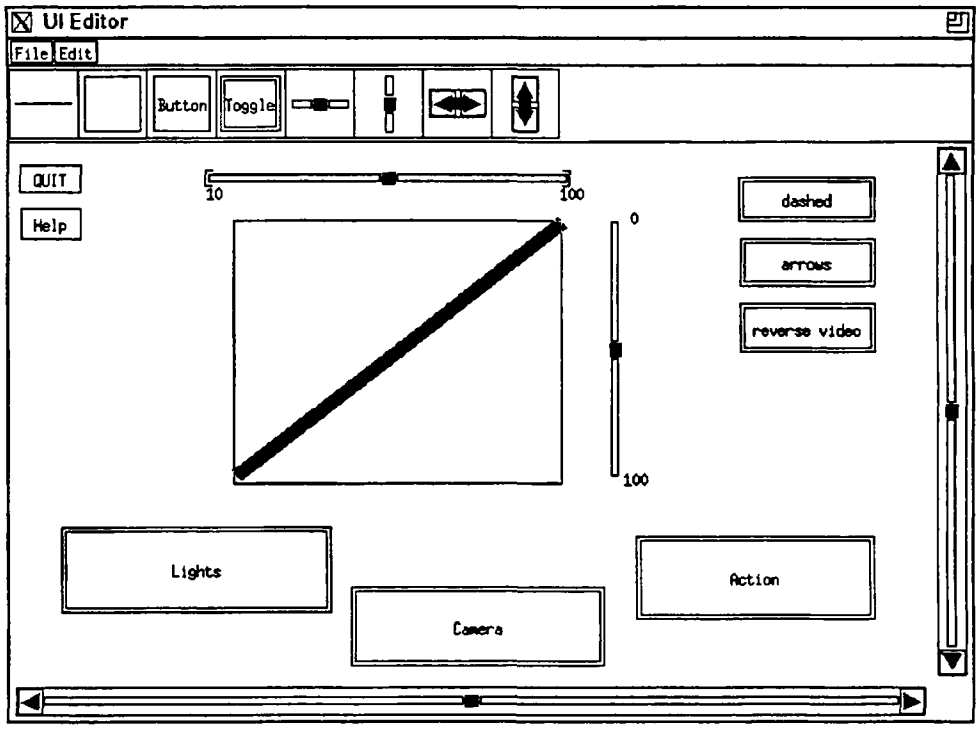


Figure 2 — X-Icon Interface Builder

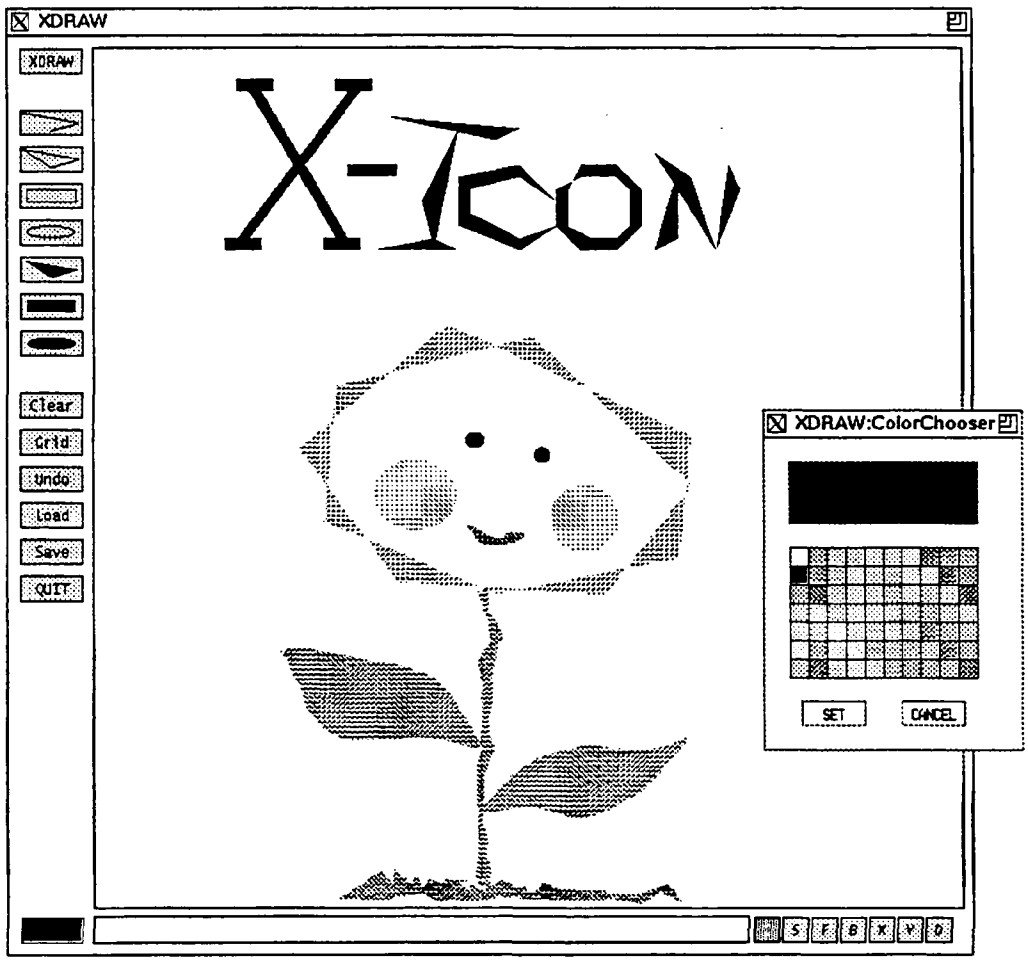


Figure 3 — Object-Oriented Drawing Program

Ordering Icon Material

What's Available

There are implementations of Icon for several personal computers, as well as for CMS, MVS, UNIX, and VMS. *Note:* Icon for personal computers requires at least 640KB of RAM; it requires more on some systems. Source code for most implementations is available.

There also is a program library that contains a large collection of Icon programs and procedures, as well as an object-oriented version of Icon that is written in Icon.

Icon Program Material

Icon programs provided by the Icon Project are in the public domain.

All program material is accompanied by documentation in printed and machine-readable form that describes how to install and use Icon. This documentation does not, however, describe the Icon programming language in detail. A book is available separately.

The current version of Icon is 8 and most programs are Version 8.0. See the note on the next page about more recent versions that are available for some platforms.

Personal Computers: Executable files, source code, and the Icon program library for personal computers are provided in separate packages. *Note:* Most material for personal computers is stored in compressed format, and most diskettes are nearly full. It therefore is necessary to have a second drive to extract the material.

CMS and MVS: The CMS and MVS packages contain executable files, source code, test programs, and the Icon program library.

UNIX: The UNIX package contains source code (but not executable files), test programs, related software, and the Icon program library. UNIX Icon can be configured for most UNIX platforms.

VMS: The VMS package contains object code, executable files, source code, test programs, and the Icon program library.

Update Subscriptions: Updates to the Icon source code and the Icon program library are available by subscription.

Source-code updates are distributed on MS-DOS diskettes in LHarc format, and are suitable for compilation under MS-DOS and OS/2 or for porting to new computers. Each update usually provides a completely new copy of the source. A source-code subscription provides five updates. Updates are issued about three times a year.

Icon program library updates are distributed on MS-DOS diskettes in plain ASCII format. A library subscription provides four updates. Updates are issued about four times a year.

Documentation

In addition to the installation guides and users' manuals included with the program packages, there are three books on Icon. One contains a complete description of the language, the second describes the implementation of Icon in detail, and the third is an introductory text designed primarily for programmers in the Humanities.

There are two newsletters. *The Icon Newsletter* contains news articles, reports from readers, information of topical interest, and so forth. It is free, and is sent automatically to anyone who places an order for Icon material. There is a nominal charge for back issues of the *Newsletter.*

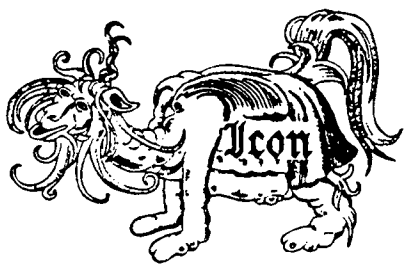
The Icon Analyst contains material of a more technical nature, including in-depth articles on programming in Icon. There is a subscription charge for the Analyst.

Payment

Payment should accompany orders and be made by check, money order, or credit card (Visa or MasterCard). The minimum credit card order is $15. Remittance *must* be in U.S. dollars, payable to The University of Arizona, and drawn on a bank with a branch in the United States. Organizations that are unable to pre-pay orders may send purchase orders, subject to approval, but there is a $5 charge for processing such orders.

Prices

The prices quoted here are good until September 30, 1992. After that, prices are subject to change without further notice. Contact the Icon Project for more current pricing information.



Versions

The symbol ↔ identifies recently released material. The symbol ● identifies Version 8.5 and 8.7 implementations. All others are 8.0.

Ordering Instructions

Media: The following symbols are used to indicate different types of media:

- ● 9-track magnetic tape
- 📼 data cartridge
- ▪ 5.25" diskette
- 💾 3.5" diskette

Tapes are written at 1600 bpi. Cartridges are written in QIC-24 format. 5.25" diskettes are 360K. 3.5" diskettes are 720/800K unless otherwise noted.

Diskettes are written in MS-DOS format except for the Amiga, the Atari ST, and the Macintosh. When ordering diskettes that are available in more than one size, specify the size (the default is shown first).

Shipping Charges: The prices listed include handling and shipping by parcel post in the United States, Canada, and Mexico. Shipment to other countries is made by air mail only, for which there are additional charges as noted in brackets following the prices. For example, the notation $15 [$5] means the item costs $15 and there is a $5 shipping charge to countries other than the United States, Canada, and Mexico. UPS and express delivery are available at cost upon request.

Ordering Codes: When filling out the order form, use the codes given in the second column of the list to the right (for example, AME, ATS, ...).

Icon Executables					
Acorn Archimedes	ARE	↔	▪ or 💾	$15	[$5]
Amiga	AME		💾	$15	[$5]
Atari ST	ATE		💾[1]	$15	[$5]
MS-DOS	DE	●	▪ or 💾	$15	[$5]
MS-DOS/386	DE-386	●	▪ or 💾	$15	[$5]
Macintosh	MET		💾	$15	[$5]
Macintosh/MPW	MEM		💾	$15	[$5]
OS/2	OE		▪ or 💾	$15	[$5]
Icon Source					
Amiga	AMS		💾	$15	[$5]
Atari ST	ATS		💾	$15	[$5]
MS-DOS	DS	●	▪ (3) or 💾 (2)	$25	[$5]
Macintosh	MST		💾	$15	[$5]
Macintosh/MPW	MSM		💾	$15	[$5]
Updates (5)	SU		▪ (2) or 💾	$60	[$15]
Icon Program Library					
Amiga	AML		💾	$15	[$5]
Atari ST	ATL		💾	$15	[$5]
MS-DOS	DL		▪ or 💾	$15	[$5]
Macintosh	ML		💾	$15	[$5]
UNIX	UL		▪ (2) or 💾	$15	[$5]
Updates (4)	LU		▪ (2) or 💾	$30	[$12]
Complete Systems					
CMS	CT		●	$30	[$10]
MVS	MT		●	$30	[$10]
UNIX	UD	↔ ●	💾[2] (3)	$25	[$5]
UNIX	UT	↔ ●	●	$30	[$10]
UNIX	UC	↔ ●	📼	$45	[$10]
VMS	VT		●	$32	[$11]
Books					
The Icon Programming Language (2nd ed.)		LB		$38	[$13]
The Implementation of Icon + update		IB		$53	[$14]
Icon Programming for Humanists + disk		HB		$36	[$10]
Newsletters					
The Icon Newsletter (all back issues, 1-38)		INC		$17	[$5]
The Icon Newsletter (back issues, each)		INS		$1	[$2[3]]
The Icon Analyst (1 year, 6 issues)		IA		$25	[$10]
The Icon Analyst (back issues, each)		IAS		$5	[$2[3]]

[1] 400K.

[2] 1.44M.

[3] Per order, regardless of the number of issues purchased.

Order Form

Icon Project • Department of Computer Science
Gould-Simpson Building • The University of Arizona • Tucson AZ 85721 U.S.A.

Ordering information: (602) 621-8448 • Fax: (602) 621-4246

name _____

address _____

_____

city _____ state _____ zipcode _____

(country) _____ telephone _____

❒ check if this is a new address

qty.	code	description	price	shipping*	total

subtotal	
sales tax (Arizona residents)	
extra shipping charges*	
purchase-order processing	
other charges	
total	

Make checks payable to The University of Arizona

The sales tax for residents of the city of Tucson is 7%.
It is 5% for all other residents of Arizona.

Payment ❒ Visa ❒ MasterCard ❒ check or money order

I hereby authorize the billing of the above order to my credit card: ($15 minimum)

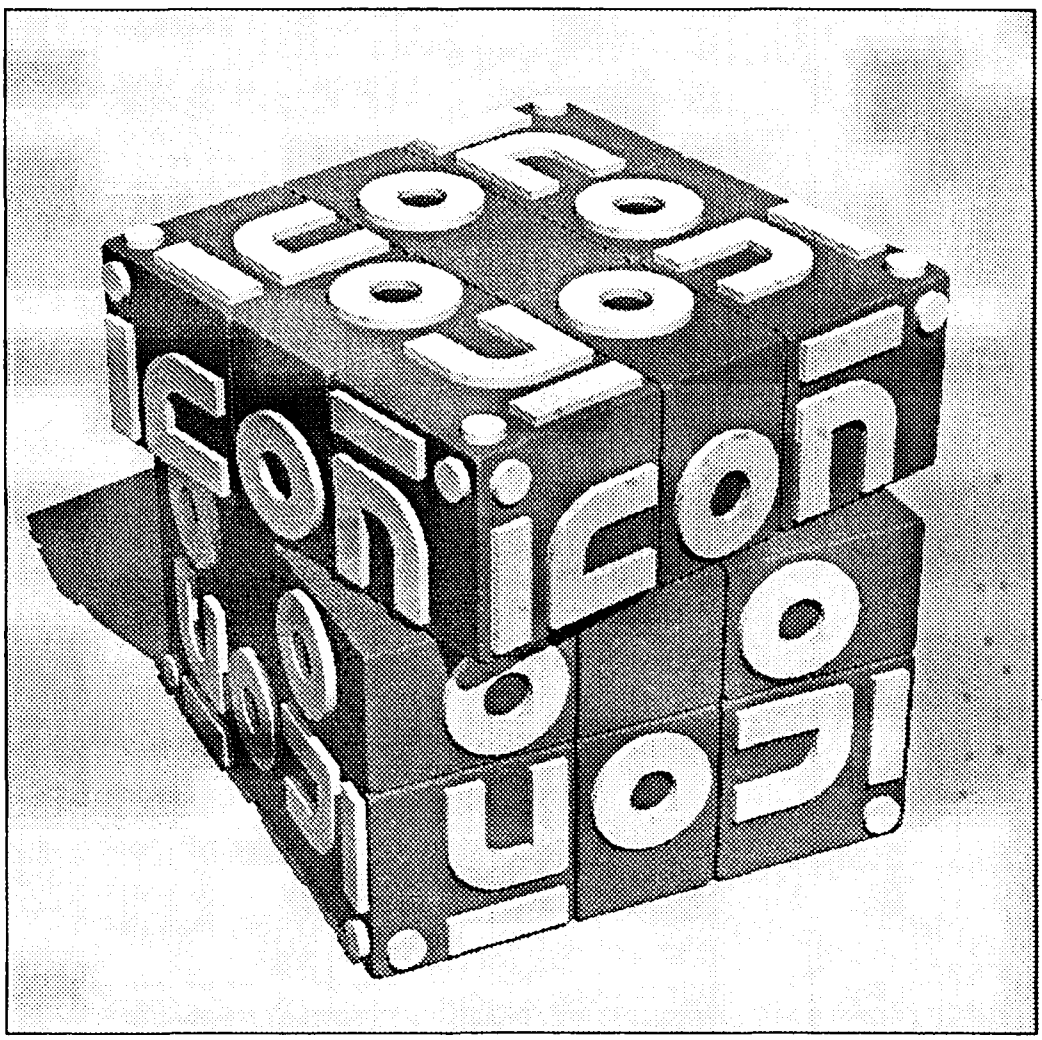
card number | | | | | | | | | | | | | | | | |

exp. date | | | | |

name on card (please print) _____

signature _____

*Shipping charges apply only to addresses outside the United States, Canada, and Mexico


FORTRAN
FORTRAN
FORTRAN