The Icon Newsletter

No. 34 – October 15, 1990

Version 8 of Icon

Version 8 of Icon is now stable and there are implementations available for all platforms previously supported.

We've sent out a lot of copies of Version 8 over the last few months and we presume most persons using Icon have upgraded to the new version. If you're still using an older version, you should seriously consider moving to Version 8. It has new features, it's faster, and there's complete, self-contained documentation for Version 8 in the second edition of *The Icon Programming Language*. The Icon Project also can no longer provide help with problems that may come up with earlier versions of Icon.

For the first time in many years, there is relatively little going on by way of implementations of Icon for new platforms. The Apple IIg, PRIME, and some models of the Cray are all we know of. Of course, part of the reason is that Icon now has been implemented on most of the computers that have the capabilities it needs.

Corrections to the Second Edition of the Icon Book

We've found a few mistakes and omissions in the second edition of *The Icon Programming Language*. These will be corrected when there's another printing, but if you have the first printing, you may wish to note the following changes:

Page 24, line 6: Change "produces the value of its left argument" to "produces the value of its right argument".

Page 129: Add the following paragraph to the end of the text, before the exercises:

> The image of an integer that has more than approximately 25 digits is given showing the approximate number of digits, as in "integer(~100)".

Page 287: Add `i1 to i2 by i3` to the list of generators at the bottom of the page.

Page 328, line 6: Change "star(string(&lcase))" to "astar(string(&lcase))".

Page 332, lines 28-30: Replace the procedure declaration for star(s) by

```
procedure astar(s)
  suspend !s | (astar(s) || !s)
end
```

Newsletters

As announced in the last *Newsletter*, we've started a new, more technical newsletter, called **The Icon Analyst**. The first two issues of the Analyst appeared in August and October. Publication is on a bi-monthly basis and the third issue is scheduled to appear in December.

The first two issues of the **Analyst** contain articles on Version 8, getting started with Icon, expression evaluation, syntactic pitfalls, memory monitoring, and timing Icon expressions. There also are programming tips and notes on advanced programming techniques.

We have upcoming articles on writing portable Icon programs, generators, large integers, memory utilization, and a series of articles on string scanning.

At the same time, the frequency of publication of *The Icon Newsletter* will drop from three times a year to twice a year. This change reflects the shift of technical content from the *Newsletter* to the **Analyst** and also the need to control costs — the *Newsletter* is now sent free of charge to over 4,600 persons.

If you're really interested in Icon — whether you're a novice or an expert — and especially if you program in Icon— you should seriously consider subscribing to the *Analyst*. Every 12-page issue is crammed with useful technical information, most of which is not available anywhere else.

New subscriptions to the *Analyst* normally start with the last published issue. We will start your subscription with an earlier issue if you wish; just make a note on the order form.

Book Prices

From time to time we get complaints about the prices of the Icon books. As we've said before, authors do not control the prices of the books they write; publishers do

We have long tried to get our publishers to set lower prices for our books. We think the prices of many books in the computer field are inflated. We also think, in addition to the cost-value question, that lower prices stimulate sales and are in everyone's best interest.

Our arguments, have, for the most part, been ignored.

Another practice we find objectionable is increasing the price of a book frequently — increases that seem to have little justification in terms of the publishers' costs. Prentice-Hall increased the price of the first edition of *The Icon Programming Language* one dollar every six months after its publication. As a result, the original $18.95 price in 1983 had risen to $31.95 by the time it was replaced by the second edition in 1990. Inflation? It surely is!

We thought we'd won a small victory when Prentice-Hall decided to publish the second edition at $29.95. This price was at least less than that of the departing first edition, and the second edition is substantially larger also. Not a bargain, maybe, but a better price than we expected.

We were, then, flabbergasted to learn that the price of the second had been increased to $32 only two weeks after its publication. It's hard to believe the new price is merely a coincidence. We suspect someone at the publisher would not tolerate an effective price reduction. We asked for an explanation in June. We have yet to get an answer.

Meanwhile, having been told the price was $29.95, we offered the book at $30 in the last *Newsletter* — rounding up a nickel to fit our format and as a token toward handling and shipping costs.

It's our policy to honor our published prices for Icon material, regardless of increased costs to us, at least until we publish a new price. So until now the Icon Project has absorbed the price increase.

Just as we went to press, we learned Prentice-Hall has increased its prices again.

As you'll see from the order form at the end of this *Newsletter*, we now place a limitation on how long our prices are guaranteed.

ICEBOL5

The first "ICEBOL" conference was held at Dakota State College in Madison, South Dakota in 1986. The primary focus of this conference was applications of the SNOBOL4 programming language to computing in the Humanities.

Subsequent conferences have expanded in scope, focusing on a broader area of interest and on other programming languages, notably Icon, Prolog and Lisp. Dakota State continues to host the conferences and now has university status.

Each conference has been better than the one before. ICEBOL4, held in October 1989, was notable for an increase in persons interested in Icon, ranging from novices to experts. The proceedings of this conference ran nearly 400 pages.

ICEBOL5, officially titled the Fifth International Conference on Symbolic and Logical Computing, will be held April 18-19, 1991. Persons who receive this *Newsletter* will get an official announcement and conference details. Persons who wish to submit papers for this conference should contact

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

electronic mail: ERIC@SDNET.BITNET

If you want more information about this conference, you can address requests as above or call (605) 256-5270.

We expect ICEBOL5 to be the best yet and the Icon Project plans to be represented. We hope to see you there.

Downloading Icon Material

Most implementations of Icon are available for downloading electronically:

BBS: (602) 621-2283

FTP: cs.arizona.edu (cd /icon)
(128.196.128.118 or 192.12.69.1)

Second Icon Workshop

In July 1988, members of the Icon Project and other persons closely associated with Icon met on the campus of Northern Arizona University in Flagstaff, Arizona. The 15 persons who attended the first workshop discussed language design, implementation, and related aspects of the Icon programming language.

A second workshop was held in July 1990 at the same site. This time 21 persons attended. The subjects discussed were similar to those of the first workshop, with new topics including the optimizing compiler for Icon and research presently underway on an Icon programming environment with tools for visualizing program performance and behavior.

Between Sessions at the Second Icon Workshop

A brief report on the second workshop is available free with any order of Icon material. Just ask for "the second workshop report" or Icon Project document IPD144.

Update on the Icon Optimizing Compiler

The Icon compiler described in the last issue of this Newsletter is now operational and most of the planned optimizations are implemented.

Preliminary timing figures for complicated programs show speed improvements in the range of 1.5 to 3 times over the Icon interpreter. Some simple programs perform much better than this under the compiler.

The optimizing compiler and the interpreter use essentially the same run-time system. The compiler deals primarily with code generation for expression evaluation, type checking, function bindings, and the allocation of space for temporary results. The compiler does not have any effect on most actual computations. For example, the compiler does not touch operations like set and table look up and there are only minor differences in storage management between the two implementations.

The optimizing compiler presently runs on a VAX under UNIX 4.3bsd. We plan to port it to other 32-bit UNIX platforms, but we have not yet set a date for possible distribution.

ProIcon Version 2.0

The ProIcon Group has announced Version 2.0 of ProIcon, an enhanced version of Icon for the Macintosh.

Version 2.0 of ProIcon includes all the features of Version 8 of Icon and well as additional facilities for working in the Macintosh environment. Notable among new features in ProIcon 2.0 is an external function interface that provides access to HyperCard XCMDs and XFCNs, and well as "stand-alone" code resources that can be used to add extensions to Icon proper.

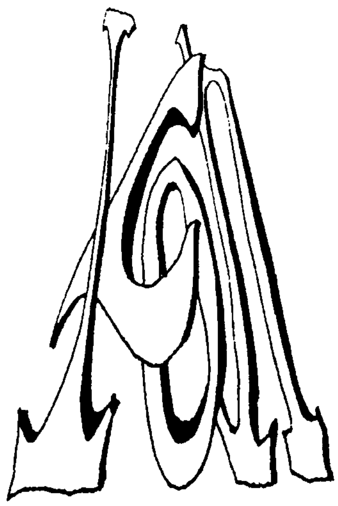
The Version 2.0 distribution also includes an application for viewing Icon allocation histories in color.

ProIcon 2.0 is scheduled to start shipping in November. The price of Version 2.0 of Icon is $175 plus shipping. Educational discounts and site licenses are available. Owners of Version 1.0 can upgrade for $35 plus shipping. Persons who purchased Version 1.0 after August 1, 1990 only pay shipping.

To order ProIcon 2.0 or get more information, contact

Catspaw, Inc.
P.O. Box 1123
Salida, CO 81201-1123
U.S.A.

(719) 539-3884



The Icon Program Library

The Icon program library consists of Icon programs and collections of procedures. The programs range from simple utilities to sophisticated text-generation applications. Most of the procedures are packages for specific applications, such as string and structure manipulation.

In addition to providing useful programs and procedures, the Icon program library also is an interesting source of examples of programming in Icon, which may be useful to persons learning Icon.

Version 8 of the Icon program library was released simultaneously with the Version 8 of Icon. The Version 8 Icon program library contains 63 complete programs and 48 sets of procedures.

Most of the material in the Icon program library is contributed by users — the work of several dozen programmers is represented in the Version 8 library.

The Icon Project collects contributions to the Icon program library, checks them to make sure they work, and packages them for distribution. Since many of the programs are complicated, it isn't practical for us to test them extensively or to certify that they work entirely as advertised. Nonetheless, many of the programs in the library are in regular use.

Subscribing to Library Updates

We continue to get many contributions to the Icon program library. We also make improvements and correct problems as they come up.

It isn't practical for us to rebuild the Icon program library every time there are changes and additions. Furthermore, the library gets larger and larger. All of this adds up to distribution problems.

We've decided to handle future distributions of the Icon program library by a series of updates to the basic Version 8 Icon program library. Updates are available by subscription, much in the manner of updates to the source code for Icon.

Updates include corrections to previously distributed program material as well as new program material. Updates are provided only on MS-DOS diskettes in ASCII text format. Both 5.25" and 3.5" diskettes are available. About four updates a year are planned, although the actual schedule will depend on available material and the resources for assembling it.

The cost for Icon program library updates is $30 for four updates, including first-class mail in the United States, Canada, and Mexico. There is an additional charge of $10 per four updates for first-class air mail delivery to other countries. Use the order form at the end of this *Newsletter* to subscribe.

Contributions to the Icon Program Library

We welcome submissions of new material to the Icon program library. But there are a few requirements. Look at the example shown on Page 5 and the accompanying remarks.

Contributed programs must be accompanied by test data and an explanation of what the results of using it should be. Test files that are read as standard input and produce results to standard output are best. For packages of procedures, a main program that links and tests them is required. In some cases, data read from standard input may not be appropriate. Send what you think is best, but realize that our resources are limited and programs that are easy to test are more likely to get into the library and to get there sooner.

(continued on page 6)

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.

(602) 621-8448

FAX: (602) 621-4246

Electronic mail may be sent to:

icon-project@cs.arizona.edu

or

...{uunet,allegra,noao}!arizona!icon-project

THE UNIVERSITY OF
ARIZONA
TUCSON ARIZONA

and

The Bright Forest Company
Tucson Arizona

Example Library Program

```
############################################################################
#
#        Name:                  deal.icn
#
#        Title:                 Deal bridge hands
#
#        Author:                Ralph E. Griswold
#
#        Date:                  June 10, 1988
#
############################################################################
```

Heading information must be in the format shown (it's processed by programs used to maintain the library).

```
#
#   This program shuffles, deals, and displays hands in the game
# of bridge.  An example of the output of deal.icn is
#
#           S: KQ987
#           H: 52
#           D: T94
#           C: T82
#
#    S: 3           S: JT4
#    H: T7          H: J9863
#    D: AKQ762      D: J85
#    C: QJ94        C: K7
#
#           S:  A652
#           H:  AKQ4
#           D:  3
#           C:  A653
#
# Options: The following options are available:
#
#    -h n Produce n hands. The default is 1.
#
#    -s n Set the seed for random generation to n.  Different seeds give different hands.
#       The default seed is 0.
#
############################################################################
```

A program description follows the heading. It should explain what the program or collection of procedures does and be sufficiently detailed so that use is clear. In the case of a program, as shown here, command-line options should be clearly evident. In the case of a collection of procedures, a prototype call for each procedure should be given with an example of the results, if appropriate.

```
#
# Links: options, shuffle
#
############################################################################
```

Requirements or restrictions, such as linked files or operating-system dependencies, should be given next in the form shown.

```
link options, shuffle

global deck, deckimage, handsize, suitsize, denom, rank, blanker
```

The code itself should begin with link, global, and record declarations.

```
procedure main(args)
  local hands, opts

  deck := deckimage := string(&letters)  # initialize global variables
  handsize := suitsize := *deck / 4
  rank := "AKQJT98765432"
  blanker := repl(" ",suitsize)
  denom := &lcase[1+:suitsize]

  opts := options(args,"h+s+")
  hands := \opts["h"] | 1
  &random := \opts["s"]

  every 1 to hands do
    display()

end
```

In the case of a program, the main procedure should appear first, followed by other procedures, preferrably in alphabetical order.

```
# Display the hands
#
procedure display()
  local layout, i
  static bar, offset
          .
          .
          .
```

Declare all local identifiers.

Icon Program Library (continued)

Complex packages complicate distribution, as do programs that read and write databases. In fact, we have several such programs that have never been distributed, simply because we haven't figured out how to put them together so that they would run on a variety of different systems. Similarly, packages that combine Icon programs with programs written in other programming languages are not likely to make it into the library.

Contributions to the Icon program library must be free of any restrictions on distribution. We do not accept copyrighted programs, regardless of any accompanying prose that appears to allow unlimited copying.

Finally, we reserve the right to decide what material to include in the library. The main considerations are quality, usefulness, compliance with the library format, and submission of adequate test data with the contribution. Submissions that duplicate functionality that already exists in the library generally will not be accepted.

All submissions are acknowledged but decisions on inclusion in the library may not be made immediately. If you've submitted material in the past that's not appeared in the library, you may wish to ask about its status — we have a lot of material from past years that lacks adequate documentation and test programs. And in some cases we've lost touch with the persons who submitted material.

Of course, we give full credit to the authors of program material that's included in the library.

Even if you don't have anything to contribute to the library, you may find a bug or make an improvement to the library. By all means let us know about such things.

FTP Access to Icon Executable Files

As mentioned in the last *Newsletter*, most Icon program material and some documentation can be obtained electronically. If you have access to FTP, you now can also get executable files for several platforms without having to download and compile the source code.

Do an anonymous FTP to cs.arizona.edu and cd to /icon/v8/binaries. There are subdirectories for different operating systems and, in these, further subdirectories for different computers. Be sure to use the binary (image) mode for file transfer of executable files.

Be aware that binary compatibility is problematical for some systems. We can't guarantee that the executable files we have in our FTP area will work as expected on other systems.

We also welcome contributions of other executable files for Version 8 of Icon. You can upload to our FTP area by doing an anonymous FTP as described above, followed by a cd to /incoming. You can then put files in the area. If you do this, send electronic mail to icon-project@cs.arizona.edu with an explanation of what you've uploaded. Material in that area is automatically removed after five days, so we need to know when it arrives.

From Our Mail

Is Icon available on CD-ROM?

No. Producing a CD-ROM master is a fairly complicated and expensive process, and different formats are required for different systems. There's not enough demand for an Icon CD-ROM yet to justify producing one.

I see Idol, the object-oriented version of Icon, is included in the Icon program library, but there's no documentation for it. How am I supposed to use it if there's no documentation?

Although we don't distribute a printed copy of the Idol user's manual with the Icon program library, there's a machine-readable version, suitable for printing, in the Idol portion of the library.

I've been experimenting with the new Version 8 functions variable(s) *and* name(v). *I thought they'd be inverses, but while* name(variable(s)) *seems always to produce* s, variable(name(v)) *doesn't produce* v *unless it's an identifier. Is this a bug?*

No, it's not a bug. It's a "feature", as they say. In the case of a structure reference like L[10], the name produced is not unique — it's just "list[10]". There's no way to tell which list it is. Granted, lists and other structures have serial numbers and you might expect the serial number to be part of the string returned by name(v), thus providing a unique identification. The problem is in the implementation of structures in Icon; there's no practical way to get the serial number of a structure from a reference to an element in it. Incidentally, variable(s) works for the names of keywords as well as identifiers.

I see The Bright Forest Company mentioned in connection with Icon from time to time. What's its relationship to the Icon Project?

The Bright Forest Company is a small, privately held corporation whose main purpose is to develop commercial implementations of Icon. Madge Griswold is president of The Bright Forest Company and Ralph Griswold is its vice president. The Bright Forest Company and Catspaw, Inc. have a joint venture, The ProIcon Group, that markets an enhanced version of Icon for the Macintosh. The Bright Forest Company also contributes to *The Icon Newsletter* and The Icon Analyst. The Bright Forest Company does not, however, receive any income from the Icon Project.

I have your two technical reports on programming in Icon, one on generators written in 1985 and one on co-expressions written in 1987. Are there any other reports in this series or any more planned?

We originally planned several reports on various aspects of programming in Icon, including ones on string scanning and programming with structures. We now are publishing material of this kind in The Icon Analyst. No further technical reports of this kind are planned.

You charge a lot of overseas air mail shipping. Wouldn't it be better to ship by surface mail unless someone is in a real rush?

Yes, overseas air mail shipping is expensive. We only charge what it costs us, on the average (rates vary somewhat from country to country with little relationship to the distance involved). The trouble with surface mail is that it is *very* slow and somewhat unreliable. We once sent a tape to India by surface mail and it took 11 months to arrive. It just isn't practical for us to deal with the problems related to surface mail (like handling inquiries about whether shipments are lost). Most of our overseas customers prefer air delivery and accept the fact that it's costly.

I was sorry to see you replaced troff versions of UNIX Icon documents with PostScript files in Version 8. I have troff but not a PostScript printer.

Distributing documentation in a form suitable for copying has always been a problem for us, since different sites have different software and printing facilities. We've stopped distributing troff source for documents because they require proprietary macro packages that many sites do not have and which we are unable to include. We realize that not all sites have access to a PostScript printer, but those that do can get high quality, correctly formatted Icon documents. There's another consideration: While we still use troff for many documents, troff is antiquated, awkward, and limited compared to current desktop publishing systems. Many of our documents (including this *Newsletter*) are now prepared using Aldus PageMaker on a Macintosh. PostScript output from this system can be distributed in the same way as PostScript for troff.

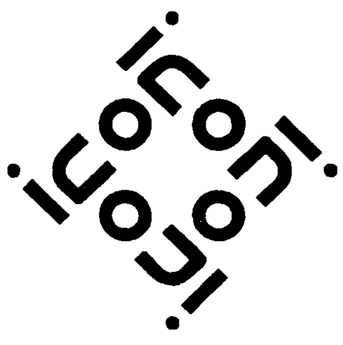
I've been wondering why all recent Icon newsletters have be exactly 12 pages long. I'd think some would be longer than others, depending on how much material you have. Or do you just wait until you have 12 pages worth? If so, doesn't this sometimes delay the newsletter?

The 12-page length of the *Newsletter* (and the Analyst) is a consequence of printing constraints. The best printing and binding method available to us at The University of Arizona uses two-up printing and saddle stitching. Two-up means that two pages are photographed at one time to make a plate for offset printing on an 11-by-17" sheet of paper. This sheet is then printed on the back with two more pages, for a total of four pages. Three of these sheets are collated, stitched, and folded to make up our 12-page, 8-1/2 by 11" newsletter. Saddle stitching is an old term for fastening the signatures at the center fold between adjacent pages. Our printer uses staples, not thread as in original saddle stitching. In any event, center stapling produces a much nicer result than corner or edge stapling — the sheets lie flat when a newsletter is opened.

It's the two-up method of printing that leads to our 12-page format. The number of pages must be a multiple of four. We usually have too much material for eight pages and not enough for 16. To be honest, we sometimes have to scramble to fill up 12 pages and still meet our publication schedule. More often, we have too much material and defer or leave out some. For example, if we hadn't had a substantial hole in this part of the *Newsletter*, we wouldn't have printed this question and answer. Graphics are handy too for making things come out right, especially since it's easy to re-size and move them in our desktop publishing system.

Incidentally, the second edition of *The Icon Programming Language* was printed 8-up, in 16-page "signatures". We produced camera-ready typeset copy for the book and in our contract the publisher stipulated exactly 384 pages, including front and end matter (24 16-page signatures). It was a bit scary writing a book that had to come out to an exact number of pages when typeset. By some kindly quirk of fate, it did come out exactly right on the first attempt, without any cutting or filling.

There! That fills the hole nicely.



An Oral History of Icon

Editors' Note: The following article was contribued by David S. Cargo, who attended the Second Icon Workshop. See the article on page 3 and the box to the right.

I had become active in the use of Icon and its promotion after reading *The Icon Programming Language* by Ralph E. Griswold and Madge T. Griswold and acquiring Icon 5.9 for MS-DOS. I used Icon to write several utilities for data base manipulation and analysis while maintaining a large mailing list.

I upgraded to later versions of Icon for MS-DOS and bought Icon for VAX/VMS for use where I worked at Honeywell Test Systems and Logistics Operations. In 1988 I informally taught a small group of people within Honeywell about Icon. Eventually, some of these people went on to write their own Icon applications.

Once I became aware of an Internet mailing list called icon-group, I joined a community of Icon users who kept in touch through electronic mail. The Internet greatly facilitates usage of Icon. It provides a medium for distribution of Icon implementations as well as a means for communicating questions and answers.

When I went to work for Cray Research in 1990, I began to work to get Icon implementations for Cray Research systems. To date, I have been able to persuade someone to develop Icon for the Cray-2. A version for Cray Y-MP will be available at an indeterminate date.

After I received an invitation to the second Icon workshop, I realized what a broad cross-section of people involved with Icon would be there. I decided that this was an opportunity to collect oral history interviews recording the history of the Icon Project.

I approached Professor Ralph Griswold and Arthur Norberg (Director of the Charles Babbage Institute) to get their permission to collect interviews under the auspicies of the Charles Babbage Institute.

After editing and review, the interviews will join the oral history collection of CBI.

The emphasis of the interviews was on the methods used to keep the Icon Project operating continuously and productively over such a long period of time. I believe the Icon Project makes a substantial and lasting contribution to research in the design and implementation of programming languages for non-numeric applications.

The Icon Project also has some parallels with other university-run projects that have produced worthwhile, useful software available at little or no cost to users. Two other examples that come readily to mind are TEX, the typesetting language produced by Donald Knuth at Stanford, and Kermit, a file transfer protocol and many implementations that are managed by Co-

The interviews at the second Icon programming language workshop were conducted for the Charles Babbage Institute. The Charles Babbage Institute is dedicated to promoting the study of the history of information processing. The institute is located on the Minneapolis campus of the University of Minnesota.

CBI was founded in 1979 to foster the research of the history of the "Information Age." Because the Information Age has developed in less than a single life-span, there is a unique opportunity to document its social, economic, and technical aspects. CBI's goal is to lead the way in collecting the knowledge and documentation possessed by key pioneering individuals and public and private institutions. Historical analysis of firsthand accounts and primary source materials provides a unique opportunity to enhance the quality of understanding of the Information Age.

CBI has several programs for promoting historical research into the history of information processing.

CBI conducts historical research itself. Topics have included development of the computer industry, scientific computation, the role of government in computing, and technical developments.

The Institute collects and makes available a collection of historical materials solicited from practitioners and organizations in the field. CBI also gathers information about collections held in other repositories in both industrial and academic archives.

The most heavily requested materials from CBI are photographs. The CBI photographic archives contain more than 3,000 photographs, including many of machines and people predating 1965.

CBI has collected oral interviews with over two hundred pioneering individuals in the information processing field. CBI staff members are continuing to collect interviews, especially with leading researchers and corporate executives who were responsible for developing strategies for their companies' involvement in information processing.

CBI has several other activities that promote research into the history of information processing. It maintains data bases that identify historically significant materials, awards fellowships to graduate students whose dissertations address aspects of the history of information processing, hosts and cosponsors conferences and lectures that provide historical perspective on information processing, and publishes a newsletter and reprints of difficult-to-obtain works of historical significance.

— *David S. Cargo*

lumbia University.

I interviewed Professor Ralph E. Griswold and Madge T. Griswold. I also interviewed four former or current graduate students who worked on Icon for Prof. Griswold: Stephen B. Wampler (now with the College of Engineering at Northern Arizona University), William H. Mitchell (now with Sunquest Information Systems), Gregg M. Townsend and Kenneth Walker (both with the Department of Computer Science at The University of Arizona). Finally, I interviewed two people who first adapted Icon to new systems, Robert E. Goldberg (with Dewar Information Systems Corporation), and Cheyenne Wills (who acted as an independent consultant for the Icon Project).

The interviews cover many of the signficant events in the history of the Icon Project: the first version of Icon in C (Stephen Wampler), the first version of Icon with detailed instructions for porting to new systems and the first version with greatly reduced requirements for assembly language support (William Mitchell), the first version of Icon for MS-DOS (Cheyenne Wills), and the first version of Icon for VAX/VMS (Robert Goldberg).

While Icon might be a tool less widely used than $\TeX$ or Kermit, the Icon Project has continually demonstrated that excellent software can come out of academic environments and earn a significant place in the real world of computer applications. I hope that collecting these oral histories will help secure a place for Icon in the history of programming languages and information processing.

An Icon Programming Environment

Work is under way at The University of Arizona on a programming environment for Icon. This is a research project that is primarily concerned with the design of high-level tools for program development, debugging, and measurement. Much of the environment will be written in Icon itself.

An enhanced version of Icon is being developed for use in this project. The two main enhancements are windowing capabilities and the ability to run several Icon programs under the same interpreter. Such programs will be able to communicate with each other.

We have chosen X-Windows for the windowing environment. This choice was made largely because the research is being done in a UNIX environment in which X has general support.

The Icon interface to X supports only a limited subset of X's capabilities. The interface is at the Xlib level, but it does not mimic Xlib precisely; rather, it is cast in a way that is natural for Icon and that utilizes the capabilities of Icon (such as generators).

One of the tools planned for the Icon programming environment is a program execution monitor. In one sense, this is just a fancy name for a debugger. We expect, however, to produce a tool with substantially greater capabilities than are found in most debuggers, so we're avoiding the "D word", lest it give the wrong impressions.

Visualization has a high priority in the tools we are developing. We want tools that can present large amounts of complex information in ways that the human mind can comprehend. This suggests an orientation toward right-brained, pattern-oriented presentations that concentrate on relationships rather than the more convention left-brained language-oriented detailed labelings and listings.

Color is an essential component of the tools we plan to develop. We require display devices that can handle at least 256 colors.

We can't show you here what we have in mind, but if you've ever watched Icon's memory-monitoring displays, you'll know what we mean.

We're just starting on this research project. Our goals are ambitious and it will take a substantial amount of time to achieve them. We'll report on our progress with the Icon programming environment from time to time in this *Newsletter*.

We also want to emphasize that this is a research project, not product development. We have no plans at present to distribute the software that we may develop.

Bug in Version 8 of Icon

We've just found a bug in Version 8 that you're unlikely to encounter, but that is still *sufficiently* serious that we're mentioning it.

The bug occurs when an empty list is formed by copying an empty list, forming an empty list section, or concatenating two empty lists. An attempt to push or put onto such a list may overwrite other data or cause a memory violation. This bug was not found for a while, since creating an empty list in one of these ways is not common.

The fix to the source code for Icon is simple, and persons who subscribe to source updates will get it when the next update is released. Updating the executable binary files we distribute is not so simple. We don't even have the capability locally to do it for all systems. So, if you're using Version 8 of Icon and your program may form a list in one of the three ways mentioned above, you need to change your code to avoid the possibility.

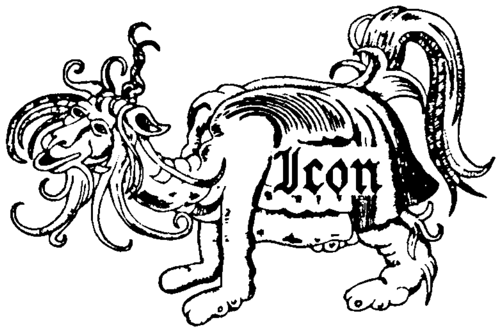
Ordering Icon Material

What's Available

There are implementations of Icon for several personal computers, as well as for CMS, MVS, UNIX, and VMS. Source code for all implementations is available. Program material is accompanied by installation instructions and users manuals in printed and machine-readable form.

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.

In addition to users manuals that are included with program material, there are three books and two newsletters about Icon.



Icon Program Material

The current version of Icon is 8. All the program material here is for Version 8.

All program material is in the public domain except the MS-DOS/386 implementation of Icon, which is a commercial product that carries a standard software license.

Personal Computers: Executables, source code, and the Icon program library for personal computers are provided in separate packages. Each package contains documentation in printed and machine-readable form. *Note:* Icon for personal computers requires at least 640KB of RAM; it requires more on some systems.

CMS and MVS: The CMS and MVS packages contain executables, source code, test programs, the Icon program library, and documentation in printed and machine-readable form.

UNIX: The UNIX package contains source code but not executables, test programs, related software, the Icon program library, and documentation in printed and machine-readable form. UNIX Icon can be configured for most UNIX systems. *Note:* executables for Xenix and the UNIX PC are available separately.

VMS: The VMS package contains object code, executables, test programs, the Icon program library, and documentation in printed and machine-readable form.

Porting: Icon source code for porting to other computers is distributed on MS-DOS format diskettes. There are two versions, one with a flat file system and one with a hierarchical file system.

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 ARC format for hierarchical file systems, and are suitable for compilation under MS-DOS 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 twice 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 December 31, 1990. After that, prices are subject to change without further notice. Contact the Icon Project for more current pricing information.

Ordering Instructions

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

●	9-track magnetic tape
📼	DC 300 XL/P cartridge
▮	360K 5.25" diskette
▤	400K 3.5" diskette
💾	800K 3.5" diskette

All cartridges are written in raw mode. All diskettes are written in MS-DOS format except for the Amiga, the Atari ST, and the Macintosh.

CMS and MVS tapes are available only at 1600 bpi. When ordering UNIX or VMS tapes, specify 1600 or 6250 bpi (1600 bpi is the default). When ordering diskettes that are available in more than one size, specify the size (5.25" is the default).

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 price. 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 fillling out the order form, use the codes given in the second column of the list to the right (for example, AME, ATE, ...).

Icon Executables

Amiga	AME	💾	$15	[$5]
Atari ST	ATE	▤	$15	[$5]
MS-DOS	DE	▮ (2) or 💾	$20	[$5]
MS-DOS/386	DE-386	▮ or 💾	$25	[$5]
Macintosh (MPW)	ME	💾	$15	[$5]
OS/2	OE	▮ or 💾	$15	[$5]
UNIX PC	UE	▮ or 💾	$15	[$5]
Xenix	XE	▮ (2) or 💾	$15	[$5]
Xenix/386	XE-386	▮ or 💾	$15	[$5]

Icon Source

Amiga	AMS	💾	$15	[$5]
Atari ST	ATS	💾	$15	[$5]
MS-DOS and OS/2	DS	▮ (2) or 💾	$20	[$5]
Macintosh (MPW)	MS	💾	$15	[$5]
Porting (flat, ASCII)	PFS	▮ (5) or 💾 (2)	$40	[$8]
Porting (hier., ARC)	PHS	▮ (2) or 💾	$20	[$5]
Source Updates (5)	SU	▮ (2) or 💾	$50	[$15]

Icon Program Library

Amiga	AML	💾	$15	[$5]
Atari ST	ATL	▤	$15	[$5]
MS-DOS and OS/2	DL	▮ or 💾	$15	[$5]
Macintosh (MPW)	ML	💾	$15	[$5]
Porting (ASCII)	PL	▮ (2) or 💾	$15	[$5]
UNIX (cpio)	UL	▮ (2) or 💾	$15	[$5]
Library Updates (4)	LU	▮ or 💾	$30	[$12]

Complete Systems

CMS	CT	●	$30	[$10]
MVS	MT	●	$30	[$10]
UNIX (cpio)	UT-C	●	$30	[$10]
UNIX (cpio)	UC-C	📼	$45	[$10]
UNIX (cpio)	UD	▮ (9) or 💾 (4)	$40	[$8]
UNIX (tar)	UT-T	●	$30	[$10]
UNIX (tar)	UC-T	📼	$45	[$10]
VMS	VT	●	$30	[$10]

Books

The Icon Programming Language (2nd ed.)	LB	$34	[$13]
The Implementation of Icon + update	IB	$45	[$14]
Icon Programming for Humanists + disk	HB	$30	[$10]

Newsletters

The Icon Newsletter (all back issues, 1-33)	INC	$15	[$5]
The Icon Newsletter (single issues, each)	INS	$1	[$0]
The Icon Analyst (1 year, 6 issues)	IA	$25	[$10]

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 ☐☐☐☐

MasterCard

name on card (please print) ______________________________

VISA

signature ______________________________

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