

The Icon Newsletter

No. 42 – July 15, 1993



Contents

- Version 8.10 of Icon ... 1
- Exploring Natural Language Syntax ... 2
- Programming Corner ... 4
- Supporting the Icon Project ... 5
- Icon Analyst* Back Issues ... 5
- From Our Mail ... 6
- Ordering Icon Material ... 9

Version 8.10 of Icon

Yes, there's another version of Icon. This one has something we think most Icon programmers really will appreciate — a built-in preprocessor. It also has some improvements to the graphic facilities ("X-Icon").

Version 8.10 presently is available for MS-DOS, OS/2, UNIX, and VMS.

The Preprocessor

The preprocessor is comparatively simple. It provides facilities for defining constants, including files, and conditional compilation. It does not, however, support macro definitions with arguments.

The syntax of preprocessor directives is simple and reminiscent of the C preprocessor. Each directive must appear on a separate line and start with a \$. Preprocessor directives can be freely intermixed with lines of Icon code.

Examples are:

```
$define Gap 2
#include "/usr/icon/lib/motif.icn"
#ifdef _MSDOS
    prompt := open("CON", "w")
#endif
```

Built-in definitions include all the implementation attributes given by &features. For example, as suggested above, a program can be specialized to a particular platform.

Graphics Features

In Version 8.10, we've added graphics functionality, eliminated some features whose effects can be obtained in other ways, and changed the way a few things are handled.

Should You Upgrade?

If you're actively using Icon, we suggest that you upgrade to Version 8.10. Once you start using the preprocessor, you'll find it simplifies programming and makes your programs easier to understand and maintain. In some cases, it will let you eliminate global identifiers and make your programs smaller. In addition, preprocessor directives will start appearing in Icon program library material.

If you're using the graphics facilities of Icon, you'll definitely want Version 8.10, since there are some incompatibilities in graphics facilities between Version 8.10 and earlier versions. Version 8.10 of the Icon program library also contains several programs and procedures that require Version 8.10 graphics facilities.

The rest of the library is unchanged from Version 8.8 (there are no preprocessor directives in the library at present).

The UNIX and VMS distribution packages contain the library, so if you get one of these, you don't need to get the library separately.

that replaces the two unsaturated nodes in the central tree by the satellite trees.

Of course, "Mary saw John" is a trivial example, and most sentences cannot be expected to work out quite as smoothly. However, for a program consisting of 1,200 lines only, FOX makes some creditable efforts. For example, it manages "What did John's sister's niece promise that she would do?", a sentence whose correct X-bar representation can keep a human parser busy for some time. However, having no semantic intuitions of any kind, FOX typically founders when the input string contains ambiguous words or floating adjuncts whose attachment options are not specified in the lexicon.

TREECAD, the third program in the package, is a didactic tool allowing an interactive guided tour of current issues in syntactic theory. The vision here is that a Linguistics 101 instructor connects his 386 portable to an overhead color LCD and clicks a friendly TREECAD icon under Windows 3.1. From a corpus file he displays a standard X-bar tree, explaining the roles and characteristics of specifiers, heads, complements and adjuncts. He demonstrates that all X-bar structures can be generated from a small set of rewrite rules, some universal, some language-specific. He may generate an abstract X-bar tree and ask his students to supply suitable terminals. He may display several subtrees and explore possible ways of combining them. Introducing the concept of govern-

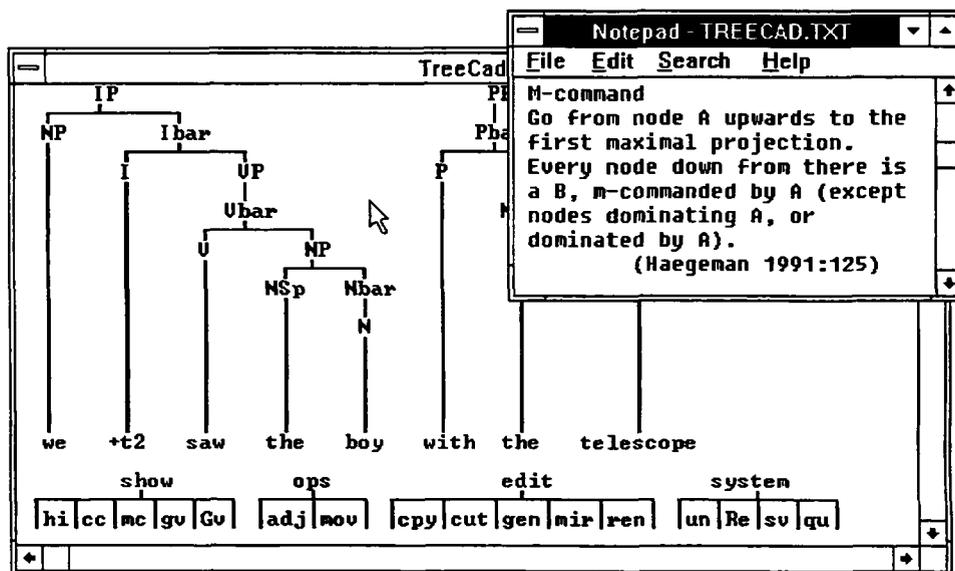
ment, he might point the mouse arrow to a node and ask, If this were a governor, which nodes would be governees? (Click: TREECAD highlights the governees.) For added flexibility, TREECAD can copy, cut, mirror, and rename structures, execute Chomsky-adjunctions and movement transformations as well as undo or redo a sequence of steps. Of course, under Windows, additional windows can be accessed (see the figure below), and all or part of the display can be exported either to a word processor or to a pixel-oriented drawing package like Paintbrush.

Subscribers to the *Icon Newsletter* can get a copy of TREE&FOX via the Icon RBBS or by FTP. Note that since FOX and TREECAD write/read directly to/from the VGA screen and make extensive use of the DOS interface functions, they can only be run from an ordinary (non-386) MS-DOS version of Icon.

References

1. The most accessible introductions to Chomsky's theories are Liliane Haegeman, *Introduction to Government and Binding Theory*, B. Blackwell, 1991, and Andrew Radford, *Transformational Grammar: A First Course*, Cambridge University Press, 1981.
2. See *The Icon Programming Language*, second edition, Chapters 14 and 15.

Editors' Note: The package mentioned above is in /icon/contrib/treefox.lzh in our FTP area and at the corresponding place on our RBBS.



TREECAD Under Windows

Programming Corner



When Icon is run from the command line, arguments are passed to the main procedure in the form of a list of strings, one for each argument. This is the main way in which information is passed to a program that is run from the command line. For example, if a program that is named `tabulate` begins with

```
procedure main(arglist)
  limit := integer(arglist[1])
  bound := integer(arglist[2])
```

and `tabulate` is called as

```
tabulate 15 30
```

`limit` is set to 15 and `bound` is set to 30. A more sophisticated program might issue an error message for a non-integer value and provide defaults for omitted arguments.

Command-line arguments can be used in any way you like. If you use a standard format that is supported by the Icon program library procedure `options()`, other Icon programmers will know how to specify options without special documentation and you can take advantage of the powerful features provided by `options()`.

`options()` was originally written by Bob Alexander. Gregg Townsend and Bob subsequently made a number of improvements. The description of `options()` that follows is based on the documentation in their program. There are more features than described here. See the program for complete documentation.

`options(arglist, optstring)` separates and interprets options given in `arglist`. Option names and values are removed from `arglist` and returned in a table.

Options are introduced by the character `-`. An option name is either a single printable character, as in `-n`, or a string of letters, as in `-geometry`. Valueless single-character options may appear in combination, for example as `-qtv`.

Some options require values. Generally, the

option name is one argument and the value appears as the next argument, as in `-F file.txt`. However, with a single-character argument name (as in this example), the value may be concatenated: `-Ffile.txt` is equivalent to the form above.

Options may be freely interspersed with non-option arguments. An argument of `-` is treated as a non-option. The special argument `--` terminates option processing. Non-option arguments are retained in the original argument list for interpretation by the caller.

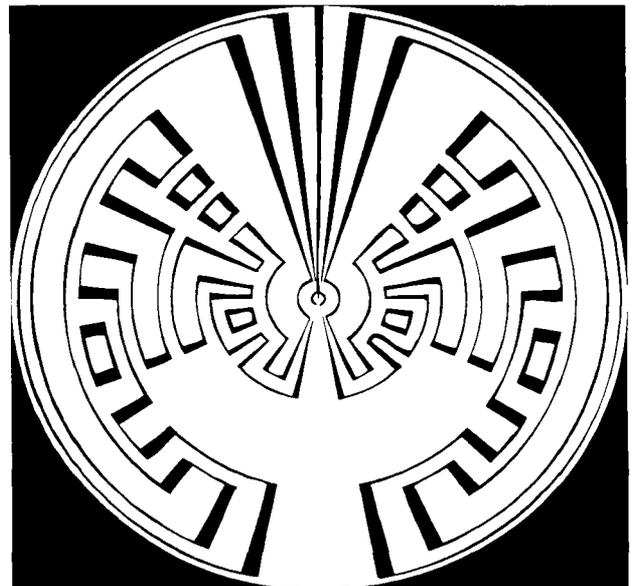
The argument `optstring` is a string specifying the allowable options. This is a concatenation, with optional spaces between one or more option specifications of the form `-name%` where `-` introduces the option name and is either a string of letters or any single printable character. `%` is one of the following flags:

- ! no value is required or allowed
- : a string value is required
- + an integer value is required
- . a real value is required

The leading `-` may be omitted for a single-character option. The `!` flag may be omitted except when needed to terminate a multi-character name. Thus, the following `optstrings` are equivalent:

```
"-n+ -t -v -q -F: -geometry: -silent"
"n+tvqF:-geometry:-silent"
"-silent!n+tvqF:-geometry:"
```

If `optstring` is omitted or null, any single letter is assumed to be valid and to require no data.



`options()` returns a table that contains the options that were specified. The keys are the specified option names. The assigned values are the data values following the options, converted to the specified type. A value of 1 is stored for options that accept no values. The table's default value is `&null`.

Upon return, the option arguments are removed from `arglist`, leaving only the non-option arguments.

Obviously, `options()` is a very capable procedure. The question is how to use it. Here's an example from the Icon program library.

The program `rsg.icn` generates randomly constructed sentences from a context-free grammar. It has three options:

- `-s i` set the seed for random numbers to `i`
- `-l i` limit sentences to at most `i` characters, default 1,000
- `-t` enable tracing

The portion of `rsg.icn` that handles these options is

```
link options
procedure main(args)
```

```
...
  opts := options(args, "s+l+t")
  &random := \opts["s"]
  limit := \opts["l"] | 1000
  trace := \opts["t"]
```

Strings from the command line are passed to `main()` in the list `args`, which in turn is passed to `options()`. The second argument to `options()` specifies that only three command-line options are allowed, and that `-s` and `-l` must have integer arguments. Thus, `options()` returns a table with entries for "s", "l", and "t". These entries are used in setting variables, as shown.

The non-null test succeeds if the option is present on the command line. Thus, `&random` is assigned a value only if `-s` appears on the command line. Similarly, `limit` is set to the value given on the command line, if `-l` appears, or to 1000, the default, if it doesn't. Finally, `trace` is assigned the value 1, provided by `options()`, if `-t` appears on the command line.

Obviously, there's some overhead in setting up a program to use `options()`. The investment usually is well worth the effort. We suggest you try it.

Supporting the Icon Project

We're very pleased with the response to our article in the last *Newsletter* about helping the Icon Project with its financial problems.

Some of you have added something extra when ordering and we've had several new subscribers to the *Analyst*, source-code updates, and program-library updates.

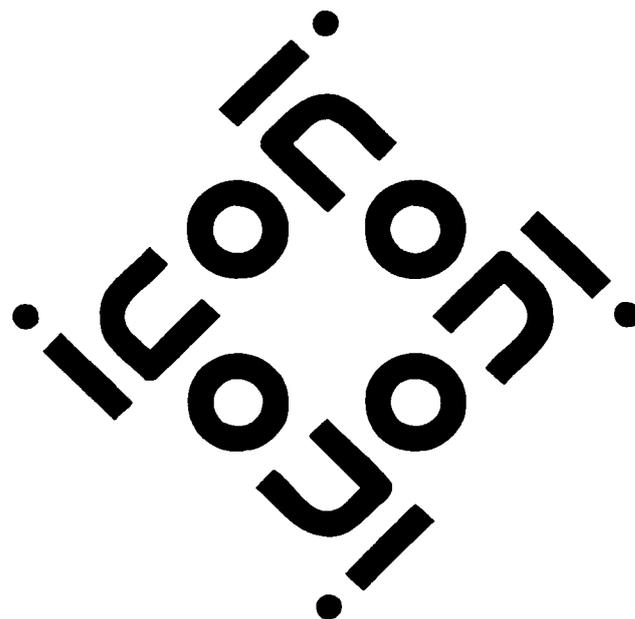
As a result, we expect to just about break even this fiscal year, which ends June 30. We expect to be able to continue to provide this *Newsletter* without charge and to continue to make improvements to Icon.

We sincerely appreciate your support and hope you find it gratifying as well.

Icon Analyst Back Issues

Several persons recently started subscribing to the *Analyst* and wanted to get back issues. With three complete years of publication already past, the cost of back issues is a bit daunting, however.

We're offering complete sets of the *Analyst* (issues 1-18) for \$60, \$15 off the per-issue price. Postage is free in the United States, Canada, and Mexico. There's a \$3 additional charge for airmail delivery to other countries. See the order form on page 10.



From Our Mail

Is the icon-group mailing list moderated?

Yes, it is. (When a news group is moderated, messages aren't sent on to group subscribers until they are approved by the moderator.) We did this because icon-group was getting duplicate messages from network news groups. We can intercept and delete messages to icon-group that we consider to be inappropriate, but so far moderation has been limited almost entirely to getting rid of duplicate messages.

Please correct the way my name appears on your mailing labels. The o has an umlaut, like this: ö. The TEX encoding is `\"o`.

We maintain our mailing list on a Macintosh. There's no problem on the Macintosh with umlauts on vowels — there are characters for these and other common letters with diacritical marks — ö, ñ, û, ç, and so forth. Our labels, however, are printed on another platform using a dot-matrix printer that doesn't know about such niceties and produces barely legible results for plain letters. We'd have to go to a laser printer to get significantly better typography. Unfortunately, labels are more expensive to produce on a laser printer. For the time being, at least, all we can do is use common transliterations like oe for ö. Incidentally, and with no apologies, we don't use TEX.

If I send you a diskette, will you copy Icon onto it for me?

No. We handle too many orders to make individual copies on user-supplied media.

Can I translate an Icon program to C using the Icon compiler?

Well, in the literal sense, you can. However, the code the Icon compiler produces is not anything you'd want to try to modify or maintain. Take a look at the output from `iconc -c` and you'll see what we mean. There are lots of reasons for the kind of C code the Icon compiler produces. For example, it must take into account generators and automatic storage management that occur in almost all Icon programs. On the other hand, it's

often possible to translate Icon programs to C by hand. (Generally speaking, it's easier to translate C programs to Icon.) A recent update to the Icon program library contains tools to help with the clerical aspects of converting code between Icon and C.

Why haven't the Amiga and Atari ST implementations of Icon been updated from 8.0?

Between Version 8.0 and the current version, 8.10, extensive changes were made in the way Icon is implemented. Updating an 8.0 implementation is a substantial undertaking, and it hasn't been done yet for the Amiga and Atari ST. We don't have the resources at the Icon Project to work on these platforms; help will have to come from elsewhere.

I'd really like to use X-Icon on my Macintosh under System 7, but there hasn't even been a mention of an implementation underway. Any hope?

Well, we wouldn't say there's no hope, but such an implementation doesn't seem likely to us. It would be a major undertaking and require considerable expert knowledge. So far, no one has even hinted they're interested in undertaking such a project.

I was impressed by the images of Icon program visualization tools that you showed in a recent issue of the Analyst. What would it take to get copies of these tools?

We're tempted to say "send money". More seriously, it's possible for us to make these tools available, but it would take a lot of work to package them for distribution. To use the tools, you'd need to build a multi-tasking version of X-Icon with monitoring instrumentation for your platform (at present, that can be done under UNIX and presumably it's possible under OS/2 and VMS). In addition, a color monitor is a virtual necessity. Despite these problems, we think the program visualization tools are exciting and we'll be working on some form of distribution. If you're interested in being involved, let us know.

Is your next programming language going to be anything like Icon?

We'll pretend we didn't hear that.

Will there be a version of Icon for NT? When?

We expect that Icon will be implemented for NT. The graphics facilities of X-Icon are particularly attractive in this context. It's too early to say when.

Does the Department of Computer Science at your university offer correspondence courses in Icon? If not, are there summer courses I could take?

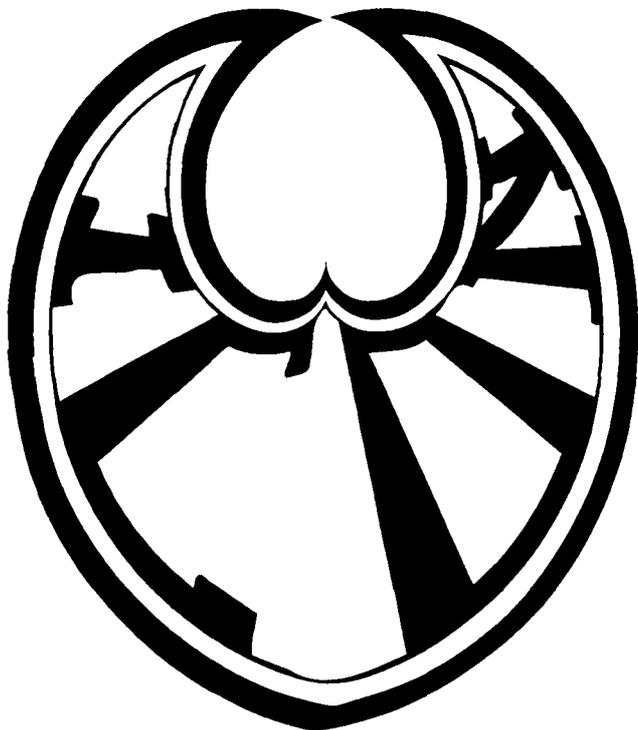
The University of Arizona doesn't offer any correspondence courses, as far as we know. The Department of Computer Science has undergraduate and graduate programs, but the courses are available only to regularly enrolled students. Icon is taught, along with several other programming languages, in an undergraduate course on comparative programming languages. Icon is the focus of a course on string and list processing that is available to both upper-division undergraduates and graduate students. Icon sometimes also is used in advanced special-topics courses, like a recent one on program visualization. Summer course offerings vary from year to year, but usually are limited to lower-division undergraduate offerings.

Is anyone building an "Icon machine" with hardware especially adapted to generators?

Not as far as we know. We hope not. If we see an advertisement for "Icon in silicon", we're running for the hills.

I heard a rumor you're writing a book on X-Icon. Anything to it?

Okay, who leaked this?



Do I have to pay the Icon Project royalties when I sell applications written in Icon?

No. However, if you wish to make a voluntary payment, it will be welcome and help support our work.

How was HOPL-II? I wanted to go but couldn't afford it.

HOPL-II (the Second History of Programming Languages Conference), was held in Cambridge, Massachusetts in April. We found it very interesting. How could it be otherwise for persons who have been involved in the design and implementation of programming languages for so many years? In addition to the formal presentations, there were lively panel discussions and, of course, the chance to meet old friends and make new ones. We also came away with a lifetime supply of anecdotes. The sessions were taped (audio but not video), the conference proceedings are being prepared, and a book is planned. We'll put a note in this *Newsletter* when we know more.

I've been laid off due to staff reductions at my company. Do you know of any jobs for Icon programmers?

There seem to be lots of programmers looking for work these days. We don't have any specific suggestions, and we honestly can't say that your Icon experience will be a selling point for you. However, we know several persons who use Icon in their jobs (sometimes on the sly) and often do well compared to their co-workers because of the edge Icon gives them.

Why aren't there articles on Icon in computer magazines?

Sometimes there are articles on Icon. The most recent one we know of was a short article by Rich Morin in the February 1993 issue of *Unix Review*. However, you're basically right — there's little about Icon in computer magazines. The editors of these magazines tend to fix on topics they perceive to be of wide interest or that are current fads. These days, topics like C++, object-oriented programming, and neural networks get most of the attention. Of course, we'd like to see more coverage of Icon. The problem is how to break through the current editorial mind set.

I've written some Icon programs that I'd like to sell, but I prefer programming to marketing. Will the Icon Project sell my programs for me? What kind of a cut would it take?

The Icon Project does not engage in commercial activities.

Many companies need someone to help design their employee benefit plans. That's my specialty and I'd like to set up an appointment to show you how I can help you.

We suppose you got our name from some mailing list. We hope your other leads are better; we don't have any employees.

Help! I have an Icon assignment due Monday and I need to print trees in an attractive way. Please send me a procedure that does this.

It sounds like you are asking us to do your school work for you. We try to be helpful, but our user support doesn't go as far as helping you cheat.

You recently mentioned some interesting files that you said are on your FTP area. I tried to get them via your RBBS but I couldn't find them.

FTP and RBBS are two separate facilities. Files for FTP reside on one of our central file servers. Our RBBS runs off a PC (literally) and its files are on the PC's hard disk. New files are first installed in our FTP area and later copied to our RBBS. There sometimes is a time lag, although it usually is only a few days. Not all of our FTP files are copied to our RBBS, however. Very large files that are impractical to download via a telephone connection generally are not kept on our RBBS. Our RBBS disk also has much less capacity than our FTP server, so some files of limited interest are available only via FTP. If you have access to electronic mail, you can get FTP files that way. See the last *Newsletter*.

Why do I have to type anonymous then I connect to your site via FTP? Since anyone can do it, why is any name needed?

Users who have accounts on our system can connect with their regular login when using FTP.

Downloading Icon Material

Most implementations of Icon are available for downloading electronically:

BBS: (602) 621-2283

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

When they do, they may get access to files that are not available to everyone. The login name anonymous is simply a way of giving anyone access to a restricted set of public files. It's just a convention and is used by all FTP sites, not just ours.

Are all these questions for real?

Yes, although we sometimes rephrase questions, combine similar questions, and provide context. Not all of the questions come from our mail, however; some come from personal conversations.

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

© 1993 by Madge T. Griswold and Ralph E. Griswold
All rights reserved.

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.

Personal Computers: Executable files and source codes are provided in separate packages. Source code for MS-DOS includes the Icon optimizing compiler, configurations for several C compilers, and also OS/2. *Note:* Personal computer distributions are 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 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 normally 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 available for MS-DOS, the Macintosh, and UNIX. A library subscription provides four updates. Updates are issued three or 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, another describes the implementation of Icon in detail, and a 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, MasterCard, or Discover). 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 August 31, 1993. After that, prices are subject to change without further notice. Contact the Icon Project for current pricing information.

Extra Payment

If you wish to support the Icon Project by making an additional payment, a line is provided at the bottom of the order form for this.

Versions

Version information is shown in parentheses. The symbol ↔ identifies recently released material.

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). In some cases, there are several diskettes in a distribution.

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.

Order 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, ...).

Executables

Acorn Archimedes (8.0)	ARE	■ or □	\$15	[\$5]
Amiga (8.0)	AME	□	\$15	[\$5]
Atari ST (8.0)	ATE	□ ¹	\$15	[\$5]
MS-DOS (8.10)	DE ↔	■ or □	\$15	[\$5]
MS-DOS 386/486 (8.10)	DE-386 ↔	■ or □	\$15	[\$5]
Macintosh (8.0)	MET	□	\$15	[\$5]
Macintosh/MPW (8.8)	MEM	□	\$15	[\$5]
OS/2 (8.10)	OE ↔	□	\$15	[\$5]

Source

Amiga (8.0)	AMS	□	\$15	[\$5]
Atari ST (8.0)	ATS	□	\$15	[\$5]
MS-DOS & OS/2 (8.10)	DS ↔	■ or □	\$30	[\$5]
Macintosh (8.0)	MST	□	\$15	[\$5]
Macintosh/MPW (8.8)	MSM	□	\$25	[\$5]
MS-DOS updates (5)	SU	■ or □	\$60	[\$15]

Complete Systems

CMS (8.0)	CT	●	\$30	[\$10]
MVS (8.0)	MT	●	\$30	[\$10]
UNIX (8.10)	UD ↔	□ ²	\$25	[\$5]
UNIX (8.10)	UT ↔	●	\$30	[\$10]
UNIX (8.10)	UC ↔	☉	\$45	[\$10]
VMS (8.10)	VT ↔	●	\$32	[\$11]

Program Library

MS-DOS (8.10)	DL ↔	■ or □	\$15	[\$5]
Macintosh (8.10)	ML ↔	□	\$15	[\$5]
UNIX (8.10)	UL ↔	■ or □	\$15	[\$5]
MS-DOS updates (4)	LU-D	■ or □	\$30	[\$12]
Macintosh updates (4)	LU-M	□	\$30	[\$12]
UNIX updates (4)	LU-U	□ ²	\$30	[\$12]

Books

<i>The Icon Programming Language</i>	LB	\$40	[\$13]
<i>The Implementation of Icon + update</i>	IB	\$53	[\$14]
<i>Icon Programming for Humanists + diskette</i>	HB	\$38	[\$10]

Newsletters

<i>The Icon Newsletter</i> (complete, 1-41)	INC	\$18	[\$5]
<i>The Icon Newsletter</i> (back issues, each)	INS	\$1	[\$2 ³]
<i>The Icon Analyst</i> (1 year, 6 issues)	IA	\$25	[\$10]
<i>The Icon Analyst</i> (complete, 1-18)	IAC	\$60	[\$3]
<i>The Icon Analyst</i> (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
	XP	Support for the Icon Project			

subtotal

Make checks payable to The University of Arizona

sales tax (Arizona residents)

The sales tax for residents of the city of Tucson is 7%.

It is 5% for all other residents of Arizona.

extra shipping charges*

purchase-order processing

other charges

total

Visa MasterCard Discover 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

