THE UNIVERSITY OF ARIZONA
TUCSON, ARIZONA 85721

DEPARTMENT OF COMPUTER SCIENCE

Icon Newsletter #21

Madge T. Griswold and Ralph E. Griswold

June 10, 1986
rec'd 7/1/86

1. Welcome, New Readers!

We've added many new names to our Icon Newsletter distribution list in recent months. There are now about 1600 persons on the list. If this is the first copy of the Icon Newsletter that you have received, welcome!

A typical issue of the Icon Newsletter includes information about new and existing implementations, reports on research projects related to Icon, a programming corner, and announcements of new publications.

The Icon Newsletter is published aperiodically, three or four times a year. At present, subscriptions are free, and we hope to maintain that policy as long as possible. However, printing and mailing costs are significant, particularly as the distribution list grows larger. At some future time it may be necessary for us to charge a nominal subscription fee.

2. Implementation News

Version 6 of Icon

Version 6 of Icon is now available for distribution. This version is written entirely in C, except for a small amount of optional assembly-language code that is needed to support co-expression context switching, and arithmetic overflow checking.

Version 6 is designed to be portable to a wide range of computers running UNIX*. To date, Version 6 has been installed successfully on the following machines:

computer	*UNIX system*
Amdahl 580	UTS
AT&T 3B2/5/15	System V
AT&T 3B20	System V
HP 9000	HP-UX
IBM PC/XT/AT	PC/IX
IBM AT	XENIX
PDP-11 (separate I & D spaces)	Version 7
Ridge 32	ROS
Sun Workstation	UNIX 4.2
VAX-11	Berkeley 4.*n*bsd

Version 6 also has been ported to the VAX-11 running VMS. This implementation, which runs on VMS Version 4.2 and higher, replaces the previous Version 5.9 implementation.

*UNIX is a trademark of AT&T Bell Laboratories.

The UNIX and VMS Icon Version 6 systems are distributed separately. Both distributions include source code and are in the public domain. The initial distribution is on magnetic tape. It is free if a tape is provided. See the distribution request form at the end of this newsletter. Copies of the UNIX system will be available on 5¼ " diskettes at a later date.

Icon for DOS

Version 5.9 of Icon is available for personal computers running MS- and PC-DOS Version 2.0 or higher. This is a small-memory-model implementation. A library of Icon programs and procedures is available on a separate diskette. See the Request Form at the end of this newsletter. This form also can be used to order copies of the book *The Icon Programming Language.*

Cheyenne Wills is working on a large-memory-model implementation of Version 6 of Icon for PCs running DOS Version 2.0 or higher. The difficulty with this implementation is that the sizes of integers and pointers are different in the large memory model. Significant changes to Icon are required to accommodate this difference. Once these changes are made, implementations for other PCs will be easier. We hope that Version 6.0 of Icon for DOS will be available this summer. A newsletter will be sent out to announce its availability.

Version 5.10 on the Sun Workstation

There are problems with Version 5.10 of Icon running on Sun Workstations under Version 3.0. While we recommend that all Sun sites switch to Version 6, the following changes should make Version 5.10 work on Sun workstations running Release 3.0.

In v5/src/link/builtin.c, lines 33-34, change

```
cmp = strcmp(btable[test], s);
if (cmp < 0)
```

to

```
if ((cmp = strcmp(btable[test], s)) < 0)
```

In v5/src/mc68000/params.h, line 16, change the value of StkBase to 0x7fffffff.

After making these changes, reinstall Icon, starting with the Icon-setup step.

3. Payment for Icon Materials

There are no fees for material related to Icon; the only charges are for the costs of media, handling, and shipping.

In cases where there are charges, payment *must* be made in U.S. dollars: our bank no longer accepts payments in other currencies because of the high cost of conversion and collection. In addition, checks must specify a bank in the United States through which payment is to be made. There is a $10 surcharge for a check written on bank that does not have a branch in the United States.

Payment should accompany orders. In the case of organizations that are unable to make prepayment, we will process purchase orders. There is, however, a $5 surcharge for this service.

Finally, we regret that we cannot accept credit-card orders.

4. Status of the Icon Workshop

Icon Newsletter #20 requested your thoughts on the possibility of an Icon Workshop, targeted for 1988, with a suggested location in Hawaii. The response to the inquiry has been relatively sparse. Among those who responded, there was a considerable range of preferences for location. Many persons commented that travel funds were difficult to obtain for locations like Hawaii.

Plans for the workshop are still unsettled. If you did not respond before, or are just learning about the projected conference now, it is not too late to express your opinion.

5. From our Mail

We get a lot of mail, much of it electronically. Here are some excepts and answers that may be of general interest.

Can I get copies of your technical reports via electronic mail?

While all of our technical reports are prepared in machine-readable form, most of them depend on formatting software and equipment that is not generally available. Consequently, we prefer to sent out printed copies.

We have a Pyramid running UNIX. Is there any hope of running Icon on it?

Until Version 6 of Icon, an implementation for the Pyramid appeared to be difficult because of the assembly language code and the unusual architecture of the Pyramid. Version 6, however, is written entirely in C and should run on the Pyramid, although it has not yet been tried. Co-expressions, however, may prove impractical to implement on the Pyramid. This would not interfere with the use of the rest of the language.

Does Icon run on the DG MV8000?

Not yet. Recently, there was an unsuccessful attempt to install Version 6 on such a DG machine. The problem is that the DG C compiler treats integer pointers and character pointers very differently, while Icon is written without regard to such distinctions.

Payment in foreign currencies is difficult for our Institute and needs several months (from 3 upto 18!!). [from France]

We appreciate the difficulties that persons in other countries have in making payments in U.S. dollars. However, we have no control over the matter. Persons in some countries use international postal money orders. If you make payment through an agency, be sure to send your order directly to us with information on how payment is made, so that we can connect your request with the order. Do not rely on an agency to attach your order to the payment.

I need handlers for UNIX signals, seek, *a window package, ...*

We get many requests for additions to Icon. Our primary function is research, and our resources for enhancing Icon are very limited. We encourage users to make additions to the language. Many of these can be handled using Icon's personalized interpreter system. At present, we have a small library of extensions (including seek). We will be happy to accept contributions of new material.

I have just recently discovered Icon on our 4.2bsd VAX and have fallen in love with it. However, I would like to make it run on a Mac. Is there source for an Icon compiler or interpreter written in Icon?

First, if you are using Icon from the Berkeley 4.2bsd distribution, you should request the current version of Icon from us. The version on the 4.2bsd tape is an old, experimental one. No, Icon is not written in Icon (Icon is written in C). Icon is not really suitable for production implementations of languages, although it is a good prototyping tool. Finally, there is not yet an implementation of Icon for the Macintosh, nor is there serious work on one under way, as far as we know. Such an implementation will be much easier once the large-memory-model modifications are completed.

I became a user of SNOBOL back in the late sixties ... . Rick has gotten Icon up on just about everything here and continues to infect almost everyone with the Icon fever — and I was especially susceptible. So far, all I miss from SNOBOL are EVAL *and* LOAD.

We're surprised that you don't miss SNOBOL's pattern matching facilities too. The usual uses of SNOBOL's EVAL function are available in Icon via co-expressions. If you mean EVAL's capability to convert strings to executable code, there is not much hope in Icon. Many Icon programmers would like to have the ability to load "external" functions during program execution in the style of SNOBOL's LOAD function. This is not out of the question, although such a facility is necessarily system dependent. We'll assign one of our research staff to this project. (That's a euphemism for "we hope someone out there will do it for us".)

I am interested in getting Icon for my organization. Please send me a schedule of fees and licensing agreements.

There is no charge for Icon. It is in the public domain and is available to anyone. There are no licenses or agreements for Icon, nor is it necessary to have a UNIX license to get Icon.

I was thinking of sending a tape for Icon, but I wonder if you would be willing to make it FTPable. It would save your people administrative and postage costs, for one thing. Not to mention much faster.

Good idea. We have set things up so that persons who have access to the ARPA Internet can get the Icon system via *anonymous* FTP to arizona.edu. To see what's available, get the file icon/README.

6. Programming Corner

Correction

In Icon Newsletter #20, there is a typographical error in the last method of printing a sorted list — the write was omitted. The correct version is:

```
a := sort(t,3)
while write(get(a),"\t",get(a))
```

Solutions to Previous Problems

1. If the default value for a table is an empty list, as in

```
t := table([ ])
```

unexpected things may happen if operations are performed on this list. It is important to understand that there is only *one* default value associated with a table. For example, if a program changes the *contents* of the default value, as in

```
while word := getword() do
   put(t[word],lineno)
```

no new elements are added to t; instead every reference to t[word] produces the default value, to which the value of lineno is added. There is never any assignment to t[word].

On the other hand, in

```
while word := getword() do
   t[word] |||:= [lineno]
```

the list concatenation operation creates a *new* list and assigns it to t[word] every time the do clause is evaluated. The first time this happens for a particular value of word, the empty list default value is concatenated with a list containing the value of lineno and this new list is assigned. The default value itself is never modified.

2. The upper bound on the number of results that find(s1, s2) can produce is *max*(*s2 – *s1 + 1, 0).

3. The difference between

```
return x
```

and

```
suspend x
fail
```

is simply an extra resumption in the second case if another result is needed in the context in which the corresponding procedure is called. This extra resumption is detectable in trace output, but otherwise does not affect program behavior. However, if the identifier x is replaced by an arbitrary expression, the two cases may produce very different results. Even if this expression itself only produces a single result, it is resumed in the second case, and this resumption may produce side effects. For example,

```
return tab(i)
```

and

```
suspend tab(i)
fail
```

may behave very differently. In the first case, assuming tab(i) itself succeeds, &pos is left set to i. In the second case, if the call is resumed, tab(i) is resumed and it restores &pos to its former value. The latter form generally is used in matching procedures to assure that scanning state variables are restored to conform to the matching protocol.

In such cases, return tab(i) would be an error. Another way to think about it is that return limits its argument to at most one result. Thus,

```
return expr
```

and

```
suspend expr \ 1
fail
```

are equivalent (except for the extra resumption).

4. The outcome of a looping expression such as

```
while expr1 do expr2
```

need not be failure. If a break expression in either *expr*$_1$ or *expr*$_2$ is evaluated, the outcome of the looping expression is the outcome of the argument of the break expression.

It is common to omit the argument of a break expression. In this case, the argument defaults to a null value. Consequently, if the break expression in

```
while expr1 do {
        .
        .
   break
        .
        .
   }
```

is evaluated, the outcome of the looping expression is the null value. In fact, if this effect is not wanted,

```
break &fail
```

can be used to assure the outcome of the looping expression is failure.

However, the argument of a break expression can be a generator. For example, if

```
break 1 to 5
```

is evaluated in a looping expression, the result sequence for the looping expression is {1, 2, 3, 4, 5}.

5. Icon programmers usually have no interest in the intermediate "ucode" that is produced by the translator to serve as input to the interpreter for Icon's virtual machine. However, getting into the internals of the implementation at this level can give insight into what goes on when an Icon program is actually executed. Hence the posed problem of finding the shortest possible Icon program whose translation contains at least one instance of every different ucode instruction.

In Version 5.10 of Icon, there are 80 different executable ucode instructions. (There also are ucode declarations, which are not of interest here.) Some of these instructions are simple stack-manipulation operations. The bulk correspond to Icon's operators — there is a different ucode instruction for every different source-language operator, except for the augmented-assignment operators.

As a start, therefore, any program that produces every different ucode instruction must have at least one instance of every operator. Beyond that, there are ucode instructions related to control structures and various specialized constructions. In the absence of a list of all the ucode instructions, they can be determined empirically. Since ucode is printable text, experimentation is easy. For example,

```
icont -c inst.icn
```

produces the ucode file inst.u1. Getting at the ucode instruction set this way is a good exercise – it illuminates aspects of Icon that few programmers ever think about.

Once all the ucode instructions are determined, the problem becomes one of finding a minimal program that produces all of them. Some things about the syntax of Icon programs can be learned by trying this.

Here is the shortest known program that produces at least one instance of every different ucode instruction under Version 5.10 of Icon (Version 6.0 has a slightly different ucode instruction set):

```
procedure y();initial|0.0[suspend(,)to"":create+-?~=!@^*./\x/x*x%x^x<x<=x=x>=x>x~=x++x---x**x||
x<<x<<=x==x>>=x>>x~==x|||x+:=x<-x:=:x<->x~===x&x.x?:=x\&pos["]]-case[]of{1:return};end
```

The program actually consists of a single 182-character line to avoid increasing its size with newline characters. We have broken it into separate lines here to fit it on the page.

Of course, if this program is executed, it immediately terminates with a run-time error message, but that is not the issue here.

To compound this lunacy, other questions can be posed. What Icon program produces the smallest *ucode* file that contains at least one instance of every different ucode instruction? Does the program above do this? Is it possible to simultaneously minimize the sizes of the Icon program and its corresponding ucode file?

As an aside, the size of a ucode file in Version 5.10 depends on the name of the file in which its source code is contained. What is the shortest file name for a source-language program that icont will accept?

Request for Icon Documents

Single copies of the documents listed below are available free of charge.

Ship to:

name _____

address _____

_____

_____

telephone _____

☐ *Dynamic Environments — A Generalization of Icon String Scanning* (TR 86-7)

☐ *Version 6 of Icon* (TR 86-10)

☐ *A Continuation Semantics for Icon Expressions* (TR 86-15)

☐ *Icon Address List* (revised)

☐ Please add the name above to the Icon mailing list.

Request for Version 6.0 of Icon for UNIX Systems

Ship to:

name _____

address _____

_____

_____

_____

telephone _____

electronic mail address _____

computers _____

operating systems _____

Tapes are written in 9-track *cpio* or *tar* format. Specify preferred format and tape recording density:

☐ cpio ☐ tar

☐ 6250 bpi ☐ 1600 bpi

Send this form, together with a magnetic tape (600′ is sufficient and preferred) *or* a check for $25 payable to The University of Arizona, to:

Icon Project
Department of Computer Science
The University of Arizona
Tucson, AZ 85721

Request for Version 6.0 of Icon for VAX/VMS

Ship to:

name _____

address _____

_____

_____

_____

telephone _____

electronic mail address _____

computer _____

operating system _____

Tapes are written in 9-track *BACKUP* format. Specify preferred tape recording density:

☐ 6250 bpi ☐ 1600 bpi

Send this form, together with a magnetic tape (600′ is sufficient and preferred) *or* a check for $25, payable to The University of Arizona, to:

Icon Project
Department of Computer Science
The University of Arizona
Tucson, AZ 85721

Request for Version 5.9 of Icon for DOS

Version 5.9 of Icon for DOS is distributed on a single 5¼″ diskette. The Icon program library, which contains a variety of application programs and collections of procedures, is available on a separate diskette. The formats available are listed below.

Ship to:

name _____

address _____

_____

_____

_____

telephone _____

electronic mail address _____

Version 5.9 of Icon:	2S/DD diskette (most PCs)	$15.00 each	_____
	2S/HD diskette (IBM AT)	$18.00 each	_____
	1S/QD diskette (DEC Rainbow)	$18.00 each	_____
Icon Program Library:	2S/DD diskette (most PCs)	$20.00 each	_____
	2S/HD diskette (IBM AT)	$23.00 each	_____
Icon Book:		$22.95 each	_____
		total	_____

Send this form, together with a check payable to The University of Arizona for the amount due, to:

Icon Project
Department of Computer Science
The University of Arizona
Tucson, AZ 85721