THE UNIVERSITY OF ARIZONA
TUCSON, ARIZONA 85721

DEPARTMENT OF COMPUTER SCIENCE

Icon Newsletter #11

Ralph E. Griswold

March 8, 1983

Icon Book

The Icon book is now in print and can be purchased though booksellers. The publication information is:

> *The Icon Programming Language*, by Ralph E. Griswold and Madge T. Griswold. Prentice-Hall, Inc. 1983. 313 pages. $19.95.

This book describes Version 5 of Icon, which is implemented in C for UNIX*. The Icon reference manual, TR 81-4a, is out of stock and is no longer distributed.

A guide to the use of the book for Version 2 users, describing the differences between the two versions, is in preparation and will be available soon.

Version 5 Implementation Information

We have completed merging the source code for the PDP-11 and VAX-11 Version 5 implementations of Icon. The resulting system, referred to as Release 5g, can be used to produce Icon on either type of computer. There are several minor improvements in this release, including production of directly executable interpreter files on the PDP-11. The procedures from the Icon book are included in this release as well. There is a request form at the end of this document for persons who want this new release.

Good progress toward a VAX/VMS implementation of Version 5 has been reported. We hope to be able to announce the availability of this system soon.

Icon Program Library

We are considering a library of useful Version 5 Icon programs, which would be included in future releases of the Icon system.

Contributions and suggestions are welcome. Any programs that are submitted must be accompanied by user documentation (for example, in UNIX man page format).

Icon Documents

In response to several requests, we have produced an up-to-date overview of Icon and an Icon bibliography. There are two new technical reports, one describing programmer-defined argument evaluation and another proposing a unification of Icon's list and string processing facilities. These documents, as well as reprints of recent papers on Icon, are available on request. Use the form provided at the end of this Newsletter.

Anyone who is interested in receiving this Newsletter is welcome to do so; just write and ask to be placed on the Icon mailing list.

*UNIX is a trademark of Bell Laboratories.

Programming Corner

Solutions to the exercises in Newsletter #10: The first exercise asked for expressions that produced certain result sequences. There are several ways of doing these, of which one set follows. Some of the parentheses are unnecessary and are included only for clarity.

(1) The squares of the positive integers: (i := 1) | |((i +:= 1) ^ 2)

(2) The factorials: (j := i := 1) | |(j *:= (i +:= 1))

(3) The Fibonacci numbers: ((i | j) := 1) | (|(i | j) := i + j)

(4) All nonempty substrings of a string s: s[(i := 1 to *s):((i + 1) to (*s + 1))]

(5) All the odd-sized substrings of s: s[(i := 1 to *s):((i + 1) to (*s + 1) by 2)]

The expression for generating the Fibonacci numbers is due to Bill Mitchell.

The second exercise turned the issue around and asked what result sequences were produced by certain expressions. The answers are:

(1) !&lcase || !&ucase: {"aA", "aB", "aC", ..., "zX", "zY", "zZ"} (26^2 strings in all)

(2) (1 to 3) + (1 to 3): {2, 3, 4, 3, 4, 5, 4, 5, 6}

(3) (1 to 3) \ (1 to 3): {1, 1, 2, 1, 2, 3} (Note that the expression that limits a result sequence can, itself, be a generator.)

(4) (1 to 5) = (4 to 9): {4, 5}

(5) 1 = |0: this is a "black hole"! It never produces a result, but its evaluation does not terminate. The reason is that the right argument produces 0, which is not equal to 1. The resulting failure cause the the right argument to be resumed. Since it is a repeated alternation, it produces 0 again, and so on. This phenomenon led to the decision to make repeated alternation terminate if its argument ever has an empty result sequence. Otherwise, for example, |read() would turn into a black hole when the end of the input file is reached. Fortunately, expressions such as the one above do not seem to occur in ordinary programming contexts. This black-hole phenomenon should not be disturbing — it is no worse than an expression such as

```
until 1 = 0
```

Limitation: In a recent letter, John Polstra commented that the limitation control structure prevents reversal of assignment in reversible-assignment expressions. For example, in

```
(x <- y) \ 1
```

the assignment to x is not reversed. This is intentional and not an implementation or design error. There are two factors involved. In the first place, the limitation control structure effectively stands between the expression it limits and the surrounding context. In this role, the limitation control structure simply limits the number of times the expression may be resumed. In the example above, the reversible assignment can be resumed only once ("resumed" as used here includes the initial evaluation, which assigns the value of y to x). On the other hand, reversal of the assignment occurs when the reversible assignment operation is resumed the second time. Although reversible assignment does not produce a second result, its resumption gives it the opportunity to reverse the assignment. The same thing is true of reversible exchange, tab(i), and move(i).

N-Queens: The non-attacking 8-queens problem is used to the point of boredom in demonstrating backtrack programming. There are solutions in the Icon Version 5 reference manual and in the Icon book. A more difficult problem is producing a program that will solve the *n*-queens problem, where *n* is a parameter. Try this one — it is not trivial.

A Puzzle: What does the following program do, and why?

```
procedure main()
   write(
      "abcde" ? {
         p() := "x"
         write(&subject)
         &subject
         }
      )
end

procedure p()
   suspend &subject[2:3]
end
```

Contacting the Icon Project

If you have a question about Icon, please feel free to contact us. U.S. mail should be sent to the address given on the forms at the end of this Newsletter. Our electronic mail addresses are:

icon-project.arizona@rand-relay	(CSNET or ARPANET)
arizona!icon-project	(Usenet)

We currently have uucp connections established through gi, mcnc, ucbvax, and utah-cs.

As of April 8, 1983, telephone numbers at the University of Arizona will change. The new numbers for contacting the Icon Project are

(602) 621-6613	department office
(602) 621-6609	Ralph E. Griswold

Request for Icon Documents

Please send the documents checked below to:

____

____

____

____

____

□ *Bibliography of the Icon Programming Language*

□ *Programmer-Defined Evaluation Regimes*, TR 82-16

□ *An Overview of the Icon Programming Language*, TR 83-3

□ *Unifying List and String Processing in Icon*, TR 83-4

□ The Evaluation of Expressions in Icon, reprinted from *ACM Transactions on Programming Languages and Systems*

□ An Implementation of Generators in C, reprinted from *Computer Languages*

□ Measuring the Performance and Behavior of Icon Programs, reprinted from *IEEE Transactions on Software Engineering*

Return this form to:

Ralph E. Griswold
Department of Computer Science
University Computer Center
The University of Arizona
Tucson, Arizona 85721
U.S.A.

Request for Release 5g of Icon for UNIX

Note: Release 5g can be configured for either PDP-11s with separate I and D spaces or VAX-11s.

Contact Information:

name: ______________________________

address: ______________________________

____

____

____

telephone: ______________________________

electronic mail address: ______________________________

cable/telex: ______________________________

All magnetic tapes are written in 9-track *tar* format.
Please specify your preferred tape recording density:

□ 1600 bpi □ 800 bpi

Return this form to: Return this form to:

Ralph E. Griswold
Department of Computer Science
University Computer Center
The University of Arizona
Tucson, Arizona 85721
U.S.A.

Enclose a magnetic tape (at least 600′) or a check for $15.00 payable to the University of Arizona.