THE UNIVERSITY OF ARIZONA

TUCSON, ARIZONA 85721

DEPARTMENT OF COMPUTER SCIENCE

Icon Newsletter #10

Ralph E. Griswold

November 8, 1982

Version 5 Icon for the VAX Operating under UNIX*

Version 5 of Icon for VAX computers operating under UNIX is now available. The system and documentation are provided free of charge; the only charge is for the cost of the tape if we are asked to provide it. There is a request form at the end of this Newsletter.

Version 5 Icon for the VAX Operating under VMS

Several persons have expressed an interest in a C implementation of Icon Version 5 for the VAX operating under VMS. If anyone is interested in undertaking such an implementation, we will provide source material and technical assistance. It should be easy, incidentally, to bring this system up under a UNIX emulator on VMS.

Porting the C Implementation of Icon

Now that the VAX implementation of Icon is running, we are working on tidying up the source code and providing more documentation. Some time this spring we should have a system ready for persons who want to transport the C implementation of Icon to other computers.

The Icon Book

The Icon book mentioned in earlier Newsletters is now in production, and is scheduled to be in stock on March 7, 1983. The title is *The Icon Programming Language* and the publisher is Prentice-Hall, Inc. More information will appear in the next Newsletter.

Electronic Mail

Interest has been expressed in creating an electronic discussion forum for Icon. If you have access to CSNET, ARPANET, or Usenet and would be interested in participating in such a group, send a note over the appropriate network to

icon−project.arizona@rand−relay (CSNET or ARPANET)

or

arizona!icon−project (Usenet)

We currently have uucp connections to gi, ucbvax, utah-cs, mcnc and purdue.

*UNIX is a trademark of Bell Laboratories.

Programming Corner

The material that follows relates to Version 5 of Icon. Most of the principles apply to Version 2 as well.

Implicit Type Conversion: In Newsletter #9, the following shuffling procedure was given:

```
procedure shuffle(x)
   every !x :=: ?x
   return x
end
```

It was noted there that x may be a string, list, table, or record, but not a cset. The reason is that the operations !x and ?x do not apply to csets directly.

If x is a cset, its value is first converted to a string and then the operation is applied. Consequently,

```
every write(!x)
```

writes the characters in the cset x as expected. However, the implicit type conversion does not change the value of x. The expression above is therefore equivalent to

```
every write(!string(x))
```

Similarly,

```
!x :=: ?x
```

is equivalent to

```
!string(x) :=: ?string(x)
```

This is much like

```
!"abc" :=: ?"abc"
```

Neither argument of the exchange operation is a variable, and a run-time error results.

Result Sequences: The result sequence for an expression in Icon consists of the results the expression is capable of producing. For example, the result sequence for 1 to 5 is {1, 2, 3, 4, 5}. The results that an expression actually produces depend on the context in which the expression is used. For example

```
every write(1 to 5)
```

causes all the results for 1 to 5 to be produced, but in

```
(1 to 5) = 2
```

only the first two results of 1 to 5 are produced.

Result sequences are interesting in themselves, independent of the context in which they are used. This subject is explored in Steve Wampler's doctoral dissertation and in the forthcoming Icon book. For example, the result sequence for

$expr_1$ | $expr_2$

is simply the result sequence for $expr_1$ followed by the result sequence for $expr_2$ (the concatenation of the result sequences).

Similarly, the result sequence for repeated alternation

|*expr*

is the repeated concatenation of the result sequences for *expr*.

From this, it follows that the result sequence for

```
(1 to 4) | (7 to 10)
```

is {1, 2, 3, 4, 7, 8, 9, 10} and the result sequence for

```
|1
```

is { 1, 1, 1, ... }, which is infinite. Similarly, the result sequence for

```
(i := 1) | |(i +:= 1)
```

is { 1, 2, 3, ... }.

An expression that fails has an empty, zero-length result sequence, {}, by definition. Empty result sequences take the place of the Boolean value *false* in control structures such as **while-do** and **if-then-else**. The empty result sequence also terminates the result sequence for repeated alternation. Thus the result sequence for

```
|read()
```

is the sequence of lines from the input file. This sequence terminates when read() fails at the end of the file.

The use of expressions that have infinite result sequences does not necessarily result in failure of the program to terminate. The generation of results from an expression can be controlled in several ways. The most direct method of controlling generation is the limitation control structure:

```
expr \ i
```

which limits *expr* to at most i results. For example, the result sequence for

```
((i := 1) | |(i +:= 1)) \ j
```

is { 1, 2, 3, ..., *j*}, assuming that the value of j is a positive integer *j*. This provides an easy way of inspecting result sequences; a typical test has the form

```
every write(expr) \ 10
```

These observations on result sequences lead to the following exercises:

1. Write expressions that have the following result sequences (do *not* use procedures):

(1) The squares of the positive integers: { 1, 4, 9, 16, ... }

(2) The factorials: { 1, 2, 6, 24, 120, ... }

(3) The Fibonacci numbers: { 1, 1, 2, 3, 5, 8, 13, ... }

(4) All nonempty substrings of a string s. For "abc" the result sequence is {"a", "ab", "abc", "bc", "c"}.

(5) All the odd-sized substrings of s.

2. What are the result sequences for the following expressions? (*Warning*: take appropriate precautions if you try to run these.)

(1) `!&lcase || !&ucase`

(2) `(1 to 3) + (1 to 3)`

(3) `(1 to 3) \ (1 to 3)`

(4) `(1 to 5) = (4 to 9)`

(5) `1 = |0`

Solutions to these exercises will appear in the next Newsletter.

Request for Version 5 Icon for VAX/UNIX

Contact Information:

name:	____________________
address:	____________________
	____________________
	____________________
	____________________
telephone:	____________________
electronic mail address:	____________________
cable/telex:	____________________

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

☐ 1600 bpi ☐ 800 bpi

Return this form to:

Ralph E. Griswold
Department of Computer Science
University Computer Center
The University of Arizona
Tucson, Arizona 85721
USA

Enclose a magnetic tape (at least 1200') or a check for $15.00 payable to the University of Arizona.