THE UNIVERSITY OF ARIZONA

TUCSON, ARIZONA 85721

DEPARTMENT OF COMPUTER SCIENCE

ICON NEWSLETTER #2

August 4, 1979

In the last six months, our work on Icon has concentrated primarily on improvements to the implementation and on distribution of systems to other locations. To date, 10 Cyber systems and seven DEC-10 systems have been sent out. The portable system has been sent to 20 individuals interested in installing Icon on other computers.

1. Version 1.3 of Icon

The current version of Icon is 1.3 (the major version number will be changed if significant changes are made to the language; the minor number indicates small changes to the language, but mainly changes to the implementation). Most of the improvements to date have been in the translator and in the runtime storage-management system. Translator parameters have been increased to allow larger programs and each Icon procedure now generates a separate Fortran subroutine. Optimizations and the introduction of "dynamic breathing room" in the storage management system have resulted in dramatic improvements (typically a factor of two to three) in the overall running speed of programs that process large amounts of data. The Cyber system now has support for the 64-character set and partial support for the NOS operating system.

Version 1.3 is available for distribution; use the attached distribution forms to obtain a copy.

2. Feedback from Users

We have received some feedback from users, although not as much as we had hoped.

Most reactions to the language itself have been favorable. The more modern syntax and traditional control structures seem to be particularly appreciated. See Section 6 for a further discussion of language features.

As we expected, much of the feedback related to problems, especially limitations in the translator, poor error messages, deficiencies in the documentation, and inefficient performance. As indicated above, we have made progress in some of these areas.

3. Implementation Issues

Performance has been a major issue. While we never expected the machine-independent, Fortran-based implementation of Icon to be particularly fast, its performance has been a disappointment. The pre-execution phases of translation and Fortran computation are particularly slow compared, for example, to SNOBOL4. Runtime speeds are not so bad, however.

To give a basis for comparison, we have written several fairly large programs involving both string and list processing in Icon and in SNOBOL4. On the DEC-10, the SITBOL implementation of SNOBOL4 runs about three times as fast as Version 1.3 of Icon, while Icon runs about twice as fast as the SIL (macro) implementation of SNOBOL4.

Since SITBOL is in wide use and is considered acceptable for some production applications, it appears that Icon's runtime performance is adequate for experimental use and that it can be brought to an acceptable level for some other applications. In any event, it is somewhat encouraging that a portable implementation of Icon is faster than a portable implementation of SNOBOL4.

Time spent prior to execution is another matter. Here SITBOL and the SIL implementation of SNOBOL4 run rings around Icon. Part of the problem with Icon lies in the language itself — the price for a language with better structure. Part of Icon's problem lies in the fact that there are two phases: translation of the Icon program into Fortran routines and then the compilation of then Fortran routines. The two phases typically take about the same amount of time, so about half of the pre-execution time can be considered as a penalty of portability (generation of Fortran code rather than relocatable binary code by the translator).

Since pre-execution processing time seriously affects program development time, especially in interactive environments, Icon is no match for SNOBOL4 in this area, especially for speculative uses and the "one-shot" programs for which SNOBOL4 is so popular. While different approaches to the implementation of Icon may make it more competitive, it remains to be seen whether it can be made to rival SNOBOL4 for such applications.

4. Portability Issues

There has been considerable enthusiasm on the part of persons for transporting Icon to a variety of machines. To date, implementations are planned or in progress for the following machines:

CDC 1784	Iris 50
CRAY 1	MELCOM-COSMO 700
Data General Eclipse	MODCOMP 4
FACOM 230-45S	PDP-11
Hitac M-180	Perkin-Elmer 8/32
Honeywell 6000	TANDEM T-16
HP 3000-III	Univac 1108
IBM/370	VAX-11/780
IBM 3031	

In some cases (notably the IBM/370), several independent implementations are in progress.

At the date of this writing, two implementations have been brought to a running state:

IBM/370:	Mr. William H. Mitchell Box 3814 16 Becton Hall North Carolina State University Raleigh, North Carolina 27650
FACOM 230-45S:	Dr. Izumi Kimura Department of Information Science Tokyo Institute of Technology Ookayama, Meguro-Ku Tokyo 152, Japan

The latter implementation has uncovered some problems with 16-bit arithmetic, but we hope to resolve these soon.

Since there are a number of implementations in progress, others will probably be running before this newsletter is distributed.

The major problems encountered by prospective porters of Icon have been with Ratfor and Fortran. Icon is written almost entirely in Ratfor and uses a number of constructs not provided by the commonly available dialects of Ratfor. Hence most porters have had to implement or modify their Ratfor systems first. Fortran limitations and idiosyncracies are presenting serious problems on some systems. Some of the newer Fortran compilers with type checking, as well as language changes in Fortran-77, are also causing problems. Although it is too early to tell for sure, it appears that the Ratfor implementation of Icon will not be as portable as we had hoped. We should have a better assessment of the situation in six months.

5. An Implementation of Icon in C

Because of the performance and portability issues raised by the Ratfor inplementation of Icon, we have begun a tentative implementation of Icon in C. Since C compilers are presently available for only a few computers, this implementation does not have a high short-range potential for actual porting. However, it will provide the basis for comparison between Fortran and C as implementation languages and, if successful, will make Icon available to the UNIX community.

6. Language Issues

We have received a number of criticisms and suggestions concerning language features in Icon. We are particularly indebted to Don Peters and Izumi Kimura for detailed and thoughtful advice.

The language issues to date fall into two categories: (1) relatively minor and specific matters and (2) major issues.

To avoid a piece-meal approach and frequent changes that are irritating to users, we are treating language changes conservatively and are particularly trying to find underlying unifying concepts. Nonetheless we are making some changes. For example, the order of local and global declarations have been fixed to allow better error detection in the translator.

Specific plans for minor changes in the future include:

1. elimination of suffix operators
2. introduction of assignment operators in the style of C
3. clarification of global scoping issues
4. replacement of the reserved word **null** by a function
5. functional notation for the creation of aggregates

A number of other changes suggested by users are being considered and will be discussed in future Newsletters.

The most interesting language issue to date is the "SNOBOL4 syndrome". It is natural for SNOBOL4 programmers to be interested in Icon and, to date, most Icon programmers have had considerable SNOBOL4 programming experience. One natural consequence of this is the tendency to transfer SNOBOL4 programming idioms to Icon. The results frequently are awkward in Icon. The typical reaction is to compare Icon unfavorably to SNOBOL4 or to suggest new features for Icon to make it resemble SNOBOL4 more closely.

A typical paradigm is the SNOBOL4 pattern

```
(P1  P2) | P3
```

The corresponding structure in Icon is

```
(e1 & e2) | e3
```

An examination of the actual use of such constructions frequently reveals that the desired effect can be obtained by

```
if e1 then e2 else e3
```

In other words, the mutual success of e1 and e2 and the backtracking implied by the alternation operation often is not actually needed. In cases when the latter construction will suffice, it is clearer, more efficient, and prevents unintended backtracking in cases where it should not occur.

In more involved cases, the "SNOBOL4 syndrome" produces Icon code that is awkward to understand and debug. Stated differently, it seems to take some time for SNOBOL4 programmers to become acculturated to Icon and to learn to use traditional control structures effectively, especially in string analysis.

On the other hand, Icon itself contains a considerable heritage from SNOBOL4, much of it unintentional and despite attempts to avoid inherited features. We suspect, therefore, that the design of Icon contains a considerable measure of the "SNOBOL4 syndrome" that has not yet been recognized. This suggests that some major work needs to be done to the language features of Icon — although, frankly, the direction is not yet clear to us.

7. Documentation

Work has been underway for some time on improvements and additions to user documentation, but help in this area is still some months off. In particular, little progress has been made on the planned program workbook.

There are two recent technical reports related to Icon design considerations and implementation:

TR 79-11: *The Design and Implementation of a Goal-Directed Programming Language*, by John T. Korb.

TR 79-12: *Icon Implementation Notes*, by David R. Hanson and Walter J. Hansen.

Copies of these reports are available for the asking.

In addition, a (somewhat out-of-date) overview of Icon appeared in the April 1979 issue of *SIGPLAN Notices*.

Upcoming papers include a general-interest paper to be presented at the ACM 79 Conference, October 29, in Detroit, and a paper on character sets and mappings to appear in the *Computer Journal*.

Acknowledgements

We are particularly indebted to Don Peters and Izumi Kimura, as noted above, for their thoughtful criticisms of Icon. Mike Shapiro has provided extensive help with adaptation of the Cyber implementation for 64-character sets and NOS implementations. Rik Smoody has called our attention to a number of deficiencies in Icon in general and the DEC-10 system in particular.

The support of the National Science Foundation under grants MCS75-01307, MCS78-02545, and MCS79-03890 has been vital to this work.

Ralph E. Griswold
David R. Hanson

THE UNIVERSITY OF ARIZONA
TUCSON, ARIZONA 85721

DEPARTMENT OF COMPUTER SCIENCE

CDC 6000/Cyber Icon Distribution Request

Contact Information

name:

address:

telephone:

cable/telex:

Computer Information

model:

memory capacity:

operating system:

character set: ☐ 63 ☐ 64

comments:

Magnetic Tape Information

Icon for CDC/Cyber systems is distributed as UPDATE PLs on an unlabeled SCOPE-format tape. Please specify your preferred tape recording charactersitics:

☐ 9-track	☐ 7-track	
☐ 1600 bpi	☐ 800 bpi	☐ 556 bpi

Please return this form with a magnetic tape (at least 1200′) to:

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

THE UNIVERSITY OF ARIZONA

TUCSON, ARIZONA 85721

DEPARTMENT OF COMPUTER SCIENCE

DEC-10 Icon Distribution Request

Contact Information

name:

address:

telephone:

cable/telex:

Computer Information

model:

memory capacity:

operating system:

comments:

Magnetic Tape Information

Icon for the DEC-10 is distibuted as a BACKUP tape in interchange mode. Please specify your preferred tape recording format:

☐ 9-track	☐ 7-track
☐ 1600 bpi	☐ 800 bpi

comments:

Please return this form with a magnetic tape (at least 1200′) to:

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

THE UNIVERSITY OF ARIZONA

TUCSON, ARIZONA 85721

DEPARTMENT OF COMPUTER SCIENCE

Portable Icon Distribution Request

Contact Information

name:	
address:	
telephone:	
cable/telex:	

Computer Information

manufacturer:	
model:	
memory capacity:	
operating system:	
Fortran pecularities:	
comments:	

Magnetic Tape Information

Our standard format for magnetic tape distribution of Icon source material is 9-track, 1600 bpi (phase encoded), EBCDIC industry standard, unlabeled, fixed-block 80-character records with a blocking factor of 10 (last block filled out to 800 characters). Please indicate if you can accept this format:

☐ yes ☐ no

If you cannot accept this format, please indicate acceptable alternatives by checking the boxes below. Circle the checks that correspond to your preferred format. (You may fill out this section even if you can accept our standard distribution format — we will try to accomodate your preferences, although doing so may cause delays.)

☐ UNIX TP tape

☐ 9-track	☐ 7-track	
☐ 1600 bpi	☐ 800 bpi	☐ 556 bpi
☐ EBCDIC	☐ ASCII	

blocking factor:

☐ 1	☐ 10	☐ other (specify)

comments:

Please return this form with a magnetic tape (at least 1200′) to:

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