The Icon Language

- Icon is a prototyping language that traces its ancestry from Pascal and SNOBOL.
- Icon is dynamically typed. It has generators, string manipulation functions, coroutines, structured data types (lists, tables, and sets), garbage collection, and built-in graphics support.
- Pick up implementations for Unix, Mac, PC, etc from ftp.cs.arizona.edu.
- With the implementation comes a huge library of useful routines and programs.
- Icon programs are usually interpreted, but there is also a compiler that translates to C.

Icon Modules

- An Icon program consists of a number of procedures declared in one or more modules. Modules are separately compiled.
- Each program must have a procedure main that will be called first when the program is started.

```
A.icn
procedure X() {
  ...
} end
procedure Y() {
  ...
} end

B.icn
procedure Z() end

M.icn
link A, B
global r, t
procedure main() {
  local s, t
  X()
  Z()
  end
```

Compiling Icon Programs

- Set these environment variables:
  ```
  setenv IPATH /usr/local/lib/icon/lib
  setenv LPATH /usr/local/lib/icon/include
  setenv FPATH /usr/local/lib/icon/bin
  ```

- To compile an Icon module M.icn do `icont -c M.icn`. This generates two files M.u1 and M.u2.

- To link an Icon program (where the main procedure is in the module M.icn) do `icont M.icn`. This generates an executable file M.

- You can pick up additional Icon programs and functions from
  ```
  /usr/local/lib/icon/lib/bipl
  /usr/local/lib/icon/lib/gipl
  ```
Procedure Declarations

- A procedure has five parts: The heading, local declarations, initializations, static declarations, and the procedure body.
- A variable that is declared static survives between procedure invocations.
- Statements in an initial clause are run the first time the procedure is called.

```plaintext
global R, T
procedure name (arguments, extra[])
    local x, y, z
    static a, b, c
    initial { ... }
    <statements>
end
```

Debugging Icon

- Bad news: There is no Icon debugger. Good news: You don’t need one!
- Since the time for an edit-compile-link is so fast, you can do your debugging using write statements.
- SETENV TRACE=-1 or &trace:=-1 will trace function calls.

Debugging Icon...

- When a runtime error occurs, execution terminates, and a traceback (a list of all active procedure calls) is generated:

```plaintext
procedure Q(); x:=x+"hello"; end
procedure P(); Q(); end
procedure main(); P(); end

Run-time error 102
File s.icn; Line 7
numeric expected
Trace back:
    main()
    P() from line 3 in s.icn
    Q() from line 2 in s.icn
    {&null + "hello"} from line 1
```

- `xdump` will display any variable type:

```plaintext
link ximage
procedure main()
    x := table(0); x[5]:="c"
    xdump([99,set([3,4]),x])
end

L2 := list(3)
L2[2] := S1 := set() insert(S1,3) insert(S1,4)
```
Confused Student Email

Question I
HI Dr. Collberg: Is there any expression in ICON similar to "&&" logical "AND" expression in PASCAL? Or should I just use:

If (true) then
  if (true) then
expr1 & expr2 succeeds (and produces expr2) if both expr1 and expr2 succeed.

Question II
Will there be questions on ICON on the final exam? No!

Question III
How do I install ICON on my [· · ·] machine? I have absolutely no idea.

Question IV

Confused Student Email...

Question V
Dear Dr. Christian:
I compile and run my program at home on my PC, transfer it to the Unix machine at the department, and then it won’t run! What’s wrong???
Sincerely,
Confused.

Dear Confused,
The .u1 and .u2 files are text files. Be sure to transfer them so that the newline characters are properly converted. Or, transfer the .icn file and recompile.

Question VI
While doesn’t this work
  every write(f2, read(f1))
while this does:
  while write(f2, read(f1))
read is not a generator.
Confused Student Email...

What could cause machcode.icn to lose track of subroutines in other files? My makefile is fine, because at one moment machcode.icn is grabbing external routines correctly then it starts randomly selecting routines to reject (i.e. &null(variables).) It’s even rejected YOUR Mcode := mcode_Create()
the second line of the first procedure!!! And then, without changing a single line of code above it, machcode will accept it again and pick some other external routine to complain about!

Icon doesn’t have a module system. In other words, all procedures are global. This is why all (most) my procedures are prefixed by the module name. What could have happened is that you’ve declared a global variable or record or procedure whose name conflicts with one of my procedures, elsewhere in the compiler. So, try to name all your global procedures/variables/records with unique (i.e. long) names.
Also, make sure that you get the case right;
mcode_Create() is different from mcode_create().

Readings and References

- http://www.cs.arizona.edu/icon/
- http://dmoz.org/Computers/Programming/Languages/Icon

The string-scanning examples were taken from
http://www.cs.arizona.edu/icon/intro.htm and

A comprehensive tutorial: