Why I'm Still Using APL

Jeffrey Shallit
School of Computer Science
University of Waterloo
Waterloo, ON N2L 3G1
Canada
www.cs.uwaterloo.ca/~shallit
shallit@cs.uwaterloo.ca

What is APL?

- a system of mathematical notation invented by Ken Iverson and popularized in his book *A Programming Language* published in 1962
- a functional programming language based on this notation, using an exotic character set (including overstruck characters) and right-to-left evaluation order
- a programming environment with support for defining, executing, and debugging functions and procedures

Frequency Distribution

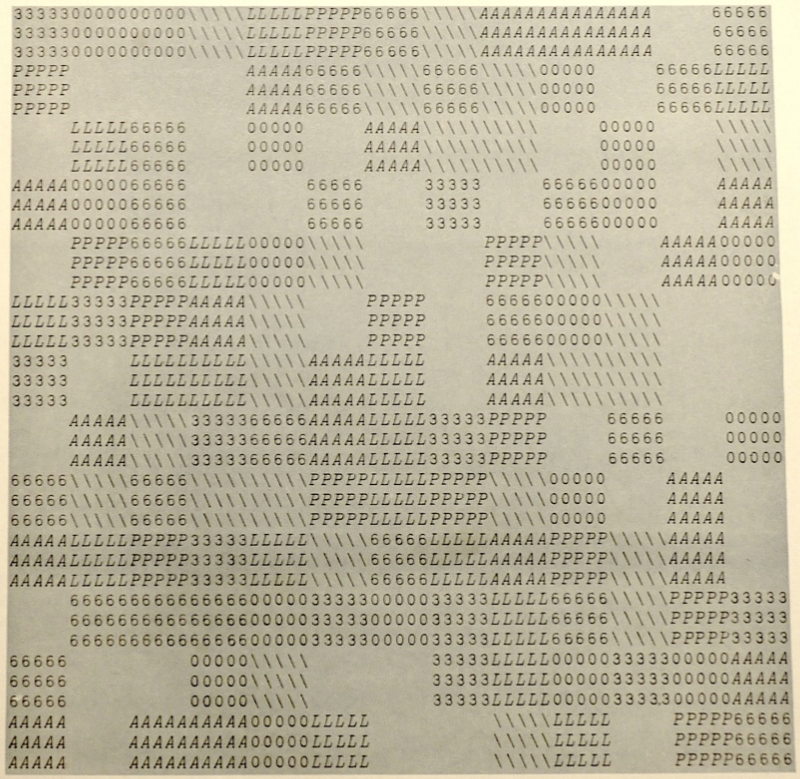
```
      ⍝ distinct elements from a list
      V←3 2 1 3 4 2 1 7 4 1 2 2 3
      ⎕←X←((⍳⍴V)=V⍳V)/V
3 2 1 4 7

      ⍝ frequency distribution
      X,[0.5] +⌿X∘.=V
3 3
2 4
1 3
4 2
7 1
```

APL and Me

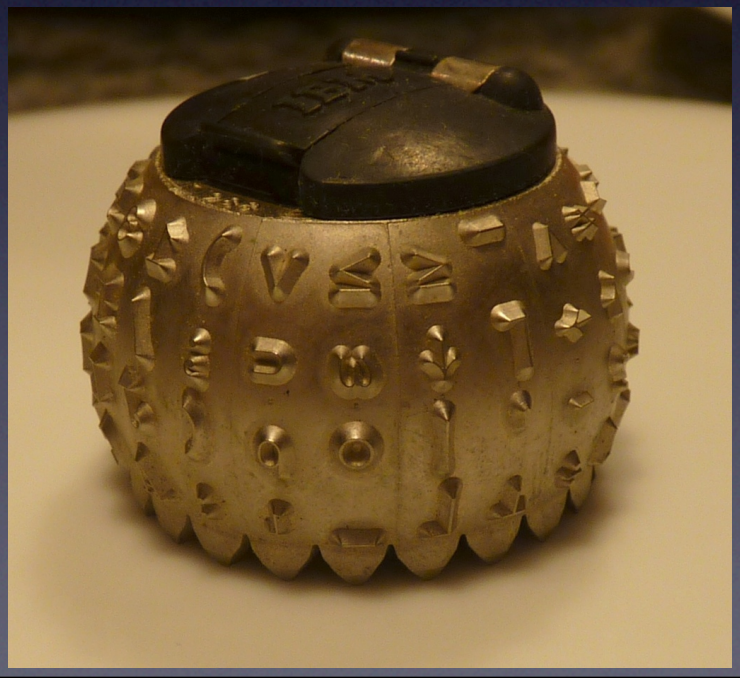
- worked at the IBM Philadelphia Scientific Center in 1973 with Ken Iverson, Adin Falkoff, Don Orth, Howard Smith, Eugene McDonnell, Richard Lathwell, George Mezei, Joey Tuttle, Paul Berry, and many others

APL\360 User's Manual



IBM Thomas J. Watson Research Center

Communicating



APL and Me

- worked as APL programmer for the Yardley Group under Alan Snyder from 1976 to 1979, with Scott Bentley, Dave Jochman, Greg Bentley, and Dave Ehret

APL and Me

- worked from 1974 to 1975 at Uni-Coll in Philadelphia as APL consultant
- worked from 1975 to 1979 as APL consultant at Princeton
- worked at I. P. Sharp in Palo Alto in 1979 with McDonnell, Berry, Tuttle, and others
- published papers at APL 80, APL 81, APL 82, APL 83 and various issues of APL Quote-Quad
- taught APL in short courses at Berkeley from 1979 to 1983 and served as APL Press representative
- taught APL at the University of Chicago from 1983 to 1988

Why I'm Still Using APL

Because

- I don't need to master arcane "ritual incantations" to do things that should be easy
- I can get stuff done *much* faster than in other programming languages
- it doesn't waste my time by making me declare the properties of data that should be obvious to anyone
- it defines edge cases correctly

Java: Read Integers and Add Them Up

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.io.StreamTokenizer;

public class SumFile {
     public static void main(String[] a) throws IOException {
      sumfile("file.txt");
  }

  static void sumfile(String filename) throws IOException {
    Reader r = new BufferedReader(new FileReader(filename));
    StreamTokenizer stok = new StreamTokenizer(r);
    stok.parseNumbers();
    double sum = 0;
    stok.nextToken();
    while (stok.ttype != StreamTokenizer.TT_EOF) {
      if (stok.ttype == StreamTokenizer.TT_NUMBER)
        sum += stok.nval;
      else
        System.out.println("Nonnumber: " + stok.sval);
      stok.nextToken();
    }
    System.out.println("The file sum is " + sum);
  }

}
```

APL: Read Integers and Add Them Up

```
      +/⍞
⍞:
      2 3 5 7 10 15
42
```

Computing Divisors of an Integer

```
      ⍝ P is a list of primes
      ⍝ E is a list of exponents

      P←2 3 7
      E←2 1 2
      P×.*E
588

      ⍝ number of divisors
      ×/E+1
18

      ⍝ list of divisors
      P×.*(E+1)⊤⍳×/E+1
1 7 49 3 21 147 2 14 98 6 42 294 4 28 196 12 84 588
       |
```

Distributing Elements along Diagonals

```
      5 6⍴⍋⍋,(⍳5)∘.+⍳6
 1  2  4  7 11 16
 3  5  8 12 17 21
 6  9 13 18 22 25
10 14 19 23 26 28
15 20 24 27 29 30
```

Guess Linear Recurrence

```
      ⍝ guess linear recurrence from data
      V←1 2 3 5 8 13 21 34
      (2↓V)⌹V[(⍳¯2+⍴V)∘.+⍳2]
1 1

      W←3 7 20 79 310 1201
      (3↓W)⌹W[(⍳¯3+⍴W)∘.+⍳3]
2 ¯1 4
```

Determining Leap Years

```
      ≠⌿0=4 100 400 ∘.| 1900 1962 2000 2012
0 0 1 1
```

Dijkstra's Question (1963)

"How would you represent a more complex operation, for example, the sum of all elements of a matrix M which are equal to the sum of the corresponding row and column indices?"

```
      M
 3 16 10 11
 5  1 14 14
19  8  6 17
 1  2 11 14
 1  8  2  9
14 12 19 17
      (,M=⊃∘.+/⍳¨⍴M)/,M
6 9
```

Design of APL

"...design decisions were made by Quaker consensus; controversial innovations were deferred until they could be revised or reevaluated so as to obtain unanimous agreement. Unanimity was not achieved without cost in time and effort, and many divergent paths were explored and assessed."

Getting Definitions Correct

```
      0*0
1

      ¯8*÷3
¯2

      +/⍳0
0

      ×/⍳0
1
```

Common Misconceptions

- APL is hard to read
- APL is hard to maintain
- APL is too slow

Dijkstra on APL

"The competent programmer is fully aware of the strictly limited size of his own skull; therefore he approaches the programming task in full humility, and among other things he avoids clever tricks like the plague. In the case of a well-known conversational programming language I have been told from various sides that as soon as a programming community is equipped with a terminal for it, a specific phenomenon occurs that even has a well-established name: it is called "the one-liners". It takes one of two different forms: one programmer places a one-line program on the desk of another and either he proudly tells what it does and adds the question "Can you code this in less symbols?" —as if this were of any conceptual relevance!— or he just asks "Guess what it does!". From this observation we must conclude that this language as a tool is an open invitation for clever tricks; and while exactly this may be the explanation for some of its appeal, viz. to those who like to show how clever they are, I am sorry, but I must regard this as one of the most damning things that can be said about a programming language."

One-Liners

$$\int_0^x (\sin t) e^{-t} dt = ??$$

$$= \frac{1}{2}\left(1 - (\cos x)e^{-x} - (\sin x)e^{-x}\right)$$

Dijkstra on APL

”APL is a mistake, carried through to perfection. It is the language of the future for the programming techniques of the past: it creates a new generation of coding bums.”

A. N. Whitehead

"By relieving the brain of all unnecessary work, a good notation sets it free to concentrate on more advanced problems and in effect increases the mental power of the race."


human designed systems, inc.
APL
POWER!
concept APL terminal
APL2
APL
IS TAKING OVER
C'est tout
réfléchi!
APL
INTERPROCESS SYSTEMS, INC.
APL 83
April 10-13, 1983
Washington, D.C.
APL
GURU
INTERPROCESS SYSTEMS, INC.
Is Data
Communications
a terminal
illness?
COMPUTERWORLD
APL
don't get no
RESPECT
INTERPROCESS SYSTEMS, INC.
A good woman
will last a year,
but
APL is forever.
INTERPROCESS SYSTEMS, INC.

Things APL Never Did Perfectly

- input and output from external files
- transfer of workspaces from one system to another
- control structures (such as "do while")
- support for novel data representations and data structures (e.g., rational numbers; extended precision, etc.)

APL's Influence

- notation for floor and ceiling are now mainstream in mathematics
- "Iverson bracket" (e.g., [x = y]) also used
- shaped modern functional programming

I'm Still Using APL

- APLX on a Macintosh
- Dyalog APL on a Sun workstation
- Experimentation, large-scale computations, research in automata theory, formal languages, and number theory, grade summaries for courses

Happy 50th Birthday to APL!