
A Brief History of PL

Babylon

- Founded roughly 4000 years ago
 - Located near the Euphrates River, 56 mi south of Baghdad, Iraq
- Historically influential in ancient western world
- Cuneiform writing system, written on clay tablets
 - Some of those tablets survive to this day
 - Those from Hammurabi dynasty (1800-1600 BC) include mathematical calculations
 - (Also known for Code of Hammurabi, an early legal code)

6

A Babylonian Algorithm

A [rectangular] cistern.
The height is 3, 20, and a volume of 27, 46, 40 has been excavated.
The length exceeds the width by 50.
You should take the reciprocal of the height, 3, 20, obtaining 18.
Multiply this by the volume, 27, 46, 40, obtaining 8, 20.
Take half of 50 and square it, obtaining 10, 25.
Add 8, 20, and you get 8, 30, 25
The square root is 2, 55.
Make two copies of this, adding [25] to the one and subtracting from the other.
You find that 3, 20 [i.e., 3 1/3] is the length and 2, 30 [i.e., 2 1/2] is the width.
This is the procedure.

– Donald E. Knuth, *Ancient Babylonian Algorithms*, CACM July 1972

The number n, m represents $n \cdot (60^k) + m \cdot (60^{k-1})$ for some k

7

More about Algorithms

- Euclid's Algorithm (Alexandria, Egypt, 300 BC)
 - Appeared in *Elements*
 - Computes gcd of two integers

```
let rec gcd a b =
    if b = 0 then a else gcd b (a mod b)
```
- Al-Khwarizmi (Baghdad, Iraq, 780-850 AD)
 - *Al-Khwarizmi Concerning the Hindu Art of Reckoning*
 - Translated into Latin (in 12th century?)
 - Author's name rendered in Latin as *algoritmi*
 - Thus the word *algorithm*

8

The Analytical Engine (Babbage)



- Charles Babbage (1791-1871, London, England)
 - Developed a mechanical calculator
 - The Difference Engine
 - Like most developers, was overly eager so during 1830's developed plans for the *Analytical Engine*
 - Never completely finished
 - But plans only discovered in 1937
 - Built in 1991 at the Science Museum of London
 - Included branching, looping, arithmetic, and storage
 - Programmed using punch cards

9

Alonzo Church (1903-1995)



- Mathematician at Princeton Univ.
- Three key contributions:
 - The lambda calculus (lectures in 1936, publ. 1941)
 - Church's Thesis
 - All effective computation is expressed by recursive (decidable) functions
 - Church's Theorem
 - First order logic is undecidable
- The modern start to PL research?
 - What is an algorithm, how do you write it down, and how much can be expressed?

10

Alan Turing (1912 - 1954)



- The father of modern computer science
 - Dissertation work advised by Church at Princeton
 - Formulated the Turing machine (~1936)
 - A formal definition of a computable algorithm

11

Early Computers



- ABC (1939-1942)
 - First electronic digital computer
 - As decided by a judge in 1973! (Invalidated ENIAC patent)
- Harvard Mark I (1944)
 - Electronic, used relays
- Z3 (1945)
 - Konrad Zuse, essentially isolated from everyone else
 - Used Plankalkül, a sophisticated programming lang.
 - But no one knew about his results, so not influential
- ENIAC (1946)
 - Electronic Numerical Integrator and Computer

12

The First Programming Languages

- Early computers could be “programmed” by rewiring them for specific applications
 - Tedious, error prone
- John von Neumann (1903-1957)
 - Three CS contributions (famous for lots of other stuff)
 - von Neumann machine – the way computers are built today
 - A **stored program architecture**
 - » Program stored in memory as data, so can be modified
 - » Without this, couldn't easily have prog. langs.
 - » Result: Programming in machine code
 - (Unclear that he actually invented this...)
 - “Conditional control transfer” – **if** and **for** statements
 - Allows for reusable code, like subroutines
 - Merge sort algorithm



13

Pseudocodes

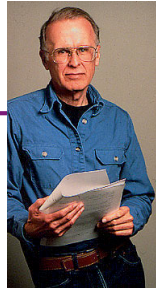
- Short Code (1949; John Mauchly)
 - Interpreted instructions
 - E.g., `X0 = sqrt(abs(Y0))` becomes `00 X0 03 20 06 Y0`
 - `06` = abs, `20` = sqrt, `03` = assignment
 - But needed to translate by hand
- A-0 Compiler (1951; Grace Murray Hopper)
 - Translated symbolic code into machine code
 - Sounds like an assembler...
 - Assigned numbers to routines stored on tape
 - Which would then be retrieved and put in memory
- Are these programming languages?



14

FORTRAN (1954 - 1957)

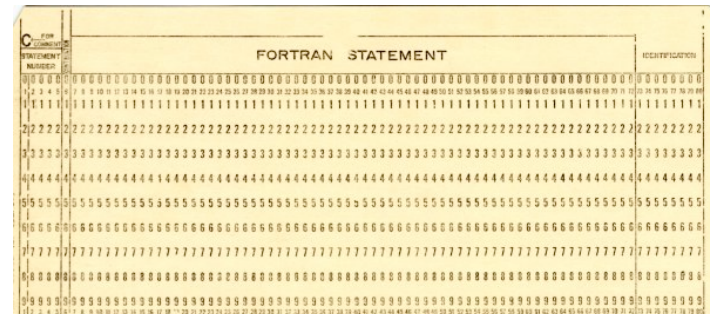
- FORMula TRANslator
- Developed at IBM by John Backus et al.
 - Aimed at scientific computation
 - Computers slow, small, unreliable
 - So FORTRAN needed to produce efficient code
- Features (FORTRAN I)
 - Variable names (up to 6 chars)
 - Loops and Arithmetic Conditionals
 - **IF (ICOUNT-1) 100, 200, 300**
 - Formatted I/O
 - Subroutines



15

Writing FORTRAN Programs

- Programs originally entered on punch cards
 - Note bevels on top-left corner for orientation
 - First five columns for comment mark or statement number
 - Each column represents one character
 - Letter: 2 punches: A=12,1 B=12,2 ..., Z=0,9



16

Punch Card Programming

- Not interactive!
 - Feed the deck into the machine
 - Or give it to someone to put in
 - Eventually get back printout with code and output
 - Could take a couple of hours if machine busy
 - Student jobs typically took overnight to run (only to find a syntax error!)
- Long test-debug cycle
 - Debugging by hand critical to not wasting time
 - Don't want to wait several hours to find you made a typo
- What happens if you drop your deck of cards?
 - Could put sequence number in corner for ordering
 - Hard to maintain this as you keep modifying program

17

Example (FORTRAN 77)

C = "comment" All-caps For loop; I goes from 2 to 12 in increments of 1

Cols 1-6 for comment or stmt label

```
C A PROGRAM TO COMPUTE MULTIPLICATION TABLES
PROGRAM TABLES
DO 20 I = 2,12
PRINT *,I,' TIMES TABLE'
DO 10 J = 1,12
10 PRINT *,I,' TIMES',J,' IS',I*J
20 CONTINUE
END
```

End of program

Source: University of Strathclyde Computer Centre, Glasgow, Scotland

18

COBOL (1959)

- COmmon Business Oriented Language
 - Project led by Hopper (again!)
- Design goals
 - Look like simple English (but doesn't read like it!)
 - Easy to use for a broad base
 - Notice: *not* aimed at scientific computing
 - Aimed at business computing instead, very successfully
- Key features
 - Macros
 - Records
 - Long names (up to 30 chars), with hyphen

19

COBOL Example (Part 1)

Program data (variables)

```
$ SET SOURCEFORMAT"FREE"
IDENTIFICATION DIVISION.
PROGRAM-ID. Iteration-If.
AUTHOR. Michael Coughlan.

DATA DIVISION.
WORKING-STORAGE SECTION.
01 Num1 PIC 9 VALUE ZEROS.
01 Num2 PIC 9 VALUE ZEROS.
01 Result PIC 99 VALUE ZEROS.
01 Operator PIC X VALUE SPACE.
```

Single-digit number

Initial value

Level number; can be used to express hierarchy (records)

Character

The source of Y2K bugs

Source: <http://www.cs.ucl.ac.uk/COBOL/examples/conditn/iterf.htm>

20

COBOL Example (Part 2)

```
PROCEDURE DIVISION.
Calculator.
  PERFORM 3 TIMES
    DISPLAY "Enter First Number      : " WITH NO ADVANCING
    ACCEPT Num1
    DISPLAY "Enter Second Number     : " WITH NO ADVANCING
    ACCEPT Num2
    DISPLAY "Enter operator (+ or *) : " WITH NO ADVANCING
    ACCEPT Operator
    IF Operator = "+" THEN
      ADD Num1, Num2 GIVING Result
    END-IF
    IF Operator = "*" THEN
      MULTIPLY Num1 BY Num2 GIVING Result
    END-IF
    DISPLAY "Result is = ", Result
  END-PERFORM.
STOP RUN.
```

Iteration (with arrow pointing to PERFORM 3 TIMES)

21

COBOL Today

- Was a DoD requirement at one time
 - Important element of its success
- Still used today
 - Legacy mainframe applications
 - New standard in 2002
 - Language has been updated for new features
 - Object-oriented?!
 - Unicode support
 - XML support

22

Discussion

- FORTRAN and COBOL were very popular
- FORTRAN, in particular, opened up a major research area: compilers!
 - A huge fraction of computer science research through the 70's, at least
 - Loads of well-developed theory and practice
- Both languages still used today
 - Yet PL researchers don't often study them...why not?

23

LISP (1958)



- LISP Processing
- Developed by John McCarthy at MIT
 - Designed for AI research
- Key ideas:
 - Symbolic expressions instead of numbers
 - Lists, lists, lists
 - Functions, functions, functions
 - Compose simpler functions to form more complex functions
 - Recursion
 - Garbage collection

24

LISP Code

```
(defun factorial (n)
  (cond ((zerop n) 1)
        (t (times n (factorial (sub1 n))))))
```

- Implemented on IBM 704 machine
 - Machine word was 36 bits
 - Two 15-bit parts, “address” and “decrement,” distinguished
 - `car` = “Contents of the Address part of Register”
 - `cdr` = “Contents of the Decrement part of Register”
- Invented `maplist` function
 - Same as `map` function in OCaml
- Used lambda notation of Church for functions

25

LISP Code as Data

- Notice that LISP programs are S-expressions
 - Which represent lists
- So LISP programs can easily manipulate LISP programs
 - Just do list operations to put programs together
 - Probably the first high-level language with this feature

26

Algol (1958)

- ALGOarithmic Language
 - Designed to be a universal language
 - For scientific computations
- Never that popular, but extremely important
 - Led to Pascal, C, C++, and Java
 - “Algol-like” languages
 - Had formal grammar (Backus-Naur Form or BNF)
 - Algol 60 added block structures for scoping and conditionals
 - Imperative language with recursive functions

27

Example Code

```
procedure Absmax(a) Size:(n, m) Result:(y)
  Subscripts:(i, k);
  value n, m; array a;
  integer n, m, i, k; real y;

  comment The absolute greatest element of the
  matrix a, of size n by m is transferred to
  y, and the subscripts of this element to i
  and k;
begin integer p, q;
  y := 0; i := k := 1;
  for p:=1 step 1 until n do
    for q:=1 step 1 until m do
      if abs(a[p, q]) > y then
        begin y := abs(a[p, q]);
              i := p; k := q;
            end
  end
end Absmax
```

Source: <http://en.wikipedia.org/wiki/ALGOL>

28

Algol 68

- Successor to Algol 60
 - But bloated and hard to use
 - And very hard to compile
 - E.g., variable names can include blanks!
- Included many important ideas
 - User-defined types
 - Code blocks that return the value of the last expr
 - struct and union
 - parallel processing (in the language)

29

Example Code

User-defined types

```
BEGIN MODE NODE = STRUCT (INT k, TREE smaller, larger),
      TREE = REF NODE;

      TREE empty tree = NIL;

      PROC add = (REF TREE root, INT k) VOID:
      IF root IS empty tree
      THEN root := HEAP NODE := (k, NIL, NIL)
      ELSE IF k < k OF root
      THEN add (smaller OF root, k)
      ELSE add (larger OF root, k)
      FI
      FI;

END
```

Identifier w/blank

Malloc

Field access

Source: <http://www.xs4all.nl/~jmvdeeer/algol68g-mk8/doc/examples/quicksort.a68.html>

30

Algol Discussion

- Good points:
 - Standard for writing algorithmic pseudocode
 - First machine-independent language
 - C, C++, Java, etc. all based on it
 - Used BNF to describe language syntax
- Bad points:
 - Never widely used; Some success in Europe
 - Hard to implement
 - FORTRAN much more popular
 - Not supported by IBM

31

Discussion

- LISP and Algol include many ideas the PL community often thinks of as “good”
 - Why so many in those two languages? Who knows?
- Yet they are not and were not “mainstream”
 - Though descendants C, C++, and Java are

32

Simula (1965)

- Developed at Norwegian Computing Center
 - By Ole-Johan Dahl and Kristen Nygaard
 - Goal was to simulate complex systems
 - Later used as a general purpose language
- Key features
 - Classes and objects
 - Inheritance and subclassing
 - Pointer to objects
 - Call by reference
 - Garbage collection
 - Concurrency via coroutines



33

Example

```
class Point(x,y); real x,y;
begin
  boolean procedure equals(p); ref(Point) p;
  if p != none then
    equals := abs(x - p.x) + abs(y - p.y) < 0.00001
  real procedure distance(p); ref(Point) p;
  if p == none then error else
    distance := sqrt((x - p.x)**2 + (y - p.y)**2);
end ***Point***

p := new Point(1.0, 2.5);
q := new Point(2.0, 3.5);
if p.distance(q) > 2 then ...
```

Parameter types

null pointer

Field access

Return value

Pointer assignment

Source: <http://www.stanford.edu/class/cs242/slides/2004/simula-smalltalk.pdf>

34

Themes

- Languages seem to move from being hardware-oriented to being task-oriented
 - Different languages are good for different things
- Syntax is important
 - Things seem to be getting more uniform
- Languages have been extremely successful
 - We are able to larger, more complex software than ever before

35

Why Care about PL Research?

(Apologies if I left off your favorite language)

- Sapir-Whorf Hypothesis: Our language influences the thoughts that we can have
 - Not likely true for human languages
 - Definitely not true for programming languages
- But language choice does affect
 - Ease of programming
 - Maintainability
 - Understandability
 - Reusability
 - Performance
 - Correctness
- And...
 - Programming languages are at the heart of CS

36