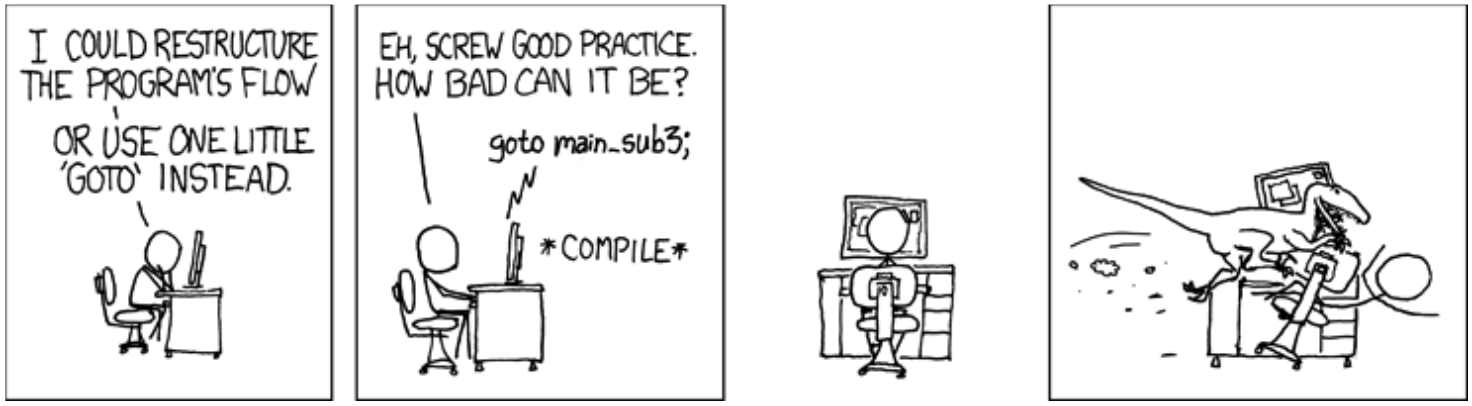
The Next 700 Programming Languages

by Eric Smith
@eric_s_smith

SVP of Product Development
Middlebury Interactive Languages



Vermont Functional Users Group - @vtfun

An Interlude

"Pay Attention to Racket - Pt. 1" - Anthony Carrico

http://vimeo.com/77754412

https://github.com/acarrico/presentations

People who like this sort of thing will find this the sort of thing they like.

- Abraham Lincoln

New Programming Languages

"By and large, the goal of introducing new programming languages has been to make it simpler to express more complex behavior."

Gul A. Agha. *Actors: A Model Of Concurrent Computation in Distributed Systems*. Ph.D Dissertation. March, 1985.

Peter J. Landin

Video of Peter Landin at Science Museum, London, UK, June, 2001 - http://vimeo.com/8955127

In Memoriam Peter Landin, Oliver Danvy at ICFP 2009, Edinburgh - http://vimeo.com/6638882

Primitives and Combinations

"Most programming languages are partly a way of expressing thing in terms of other things and partly a basic set of given things. ... many linguistic idiosyncracies are concerned with the former ... whereas aptitude for a particular class of tasks is essentially determined by the latter"

Peter J. Landin. *The Next 700 Programming Languages*. CACM, Vol. 9, No 3. March, 1966.

ISWIM (If you See What I Mean)

"ISWIM is an attempt at a general purpose system for describing things in terms of other things, that can be problem-oriented by appropriate choice of 'primitives'."

Peter J. Landin. *The Next 700 Programming Languages*. CACM, Vol. 9, No 3. March, 1966.

Influencing language evolution

"... do the idiosyncracies [of programming languages] reflect basic logical properties of the situations they are being catered for? Or are they accidents of history and personal background that may be obscuring fruitful developments. This question is clearly important if we are trying to predict or influence language evolution."

Peter J. Landin. *The Next 700 Programming Languages*. CACM, Vol. 9, No 3. March, 1966.

From a well-mapped space

"... we must think in terms, not of languages, but of families of languages. ... we must systematize their design so that a new language is a point chosen from a well-mapped spaced"

Peter J. Landin. *The Next 700 Programming Languages*. CACM, Vol. 9, No 3. March, 1966.

Timeline

- Euclid - *Elements* (2300 years ago). Theorems built from axioms using logical proofs.
- Newton - *Principia* (1687). Propositions using geometrical proofs.
- Gottfried Leibnitz - 1684. His notation for calculus is still used today.
- Gottlob Frege - *Begriffschrift* (1879). Predicate logic
- Whitehead & Russel - *Principia Mathematica*, started in 1910. Attempt at axiomatic foundation for mathematics
- David Hilbert - 1900. Second problem: Prove that arithmetic is free of internal contradictions.

Timeline (cont.)

- David Hilbert - *Grundlagen* (1928). Entscheidungsproblem: Is there a procedure to determine if a statement (in logic) is provable?
- Kurt Gödel - *On formally undecidable propositions* (1931). Incompleteness theorem: For any consistent system of axioms about the natural numbers, there will be statements that are true, but unprovable.
- Alonzo Church - *An unsolvable problem of elementary number theory* (1936). Answer to Entscheidungsproblem is 'no'. Re-used his theory of functions in his proof.
- Alan Turing - *On Computable Numbers* (1936). Proof for Entscheidungsproblem using hypothetical machine. Is there a general way of determining if the machine will stop? Universal machine.

Timeline (cont.)

- Alonzo Church - *The Calculi of Lambda Conversion* (1941).
- Von Neumann - *First Draft of a Report on the EDVAC* (1945). Leaked report describes instructions and data stored in the same way.
- Manchester "Baby" (1948). First stored-program computer in operation.
- FORTRAN (1954, rel. 1957). Created for the IBM 704.
- John McCarthy - *Recursive Functions of Symbolic Expressions* (1960).
- Mervyn Pragnell - (late 1950s). Reading group, including Peter Landin, Christopher Strachey, Rod Burstall, Dana Scott, Robin Milner, et al

Timeline (cont.)

- Peter Landin - *The Mechanical Evaluation of Expressions* (1964). Evaluating lambda expressions using the SECD 'machine'.

- Peter Landin - *A Correspondence Between ALGOL 60 and Church's Lambda-Notation* (1965). Mapping expressions in ALGOL 60 to lambda expressions.

- Peter Landin - *The Next 700 Programming Languages* (1966). Describes a language 'family' with lambda expressions at the core, plus mutable variables and J-operator.

λ terms

t ::= x | λx.t | tt

Functions and λ-expressions

Abstraction

```
f(x) = 10x² + 4x + 3

λx.10x² + 4x + 3
```

Application

```
f(5)

(λx.10x² + 4x + 3)5
```

Features of λ calculus

- In pure lambda calculus, everything is a function.
- Data and data structures (including constructors and selectors) can be represented using only λ-expressions.
- Conditional branching using λ-expressions.
- Functions can be recursive using the Y-combinator
- Take functions as arguments.
- Return functions.

"Church without Lambda"

```
(λx.10x² + 4x + 3)5
```

where-expression

```
10x² + 4x + 3 where x = 5
```

```
10x² + 4x + 3
where x = 5
```

let expression

```
let x = 5; 10x² + 4x + 3
```

```
let x = 5
10x² + 4x + 3
```

The offside rule

"The only indentation rule used to resolve ambiguities is that the whole subexpression lies in the southeast quadrant of the page defined by its first character."

William H. Burge. *Recursive Programming Techniques*. The Systems Programming Series. 1975.

Miranda

```
f x = g y z
      where
      y = (x+1)*(x-1)
      z = p x (q y)
g r = groo (r+1)
```

SECD machine

- **Stack** which is a list, each of whose items is an intermediate result of evaluation, awaiting subsequent use.
- **Environment** which is a list-structure made up of name/value pairs.
- **Control** which is a list, each of whose items is an AE awaiting evaluation, ...
- **Dump** which is a complete state, i.e. comprising four components as listed here.

Free variable

A particular instance of a variable is “free” in a lambda expression if it is not “bound” by a lambda

```
λxy.xyz
```

Trick question

```
λy. x (λx. x x) x
```

Closure

"We represent the value of a λ-expression by a bundle of information called a "closure," comprising the λ-expression and the environment relative to which it was evaluated."

Peter J. Landin. *Mechanical evaluation of expressions*. The Computer Journal (British Computer Society). 1964.

```
f = λx.λy.x+y
f x = λy.x+y
g = f 10
g 20

=> 30
```

Imperative AE

"The notion of AE is extended to comprise one new format that models assignment."

Peter J. Landin. *A Correspondence Between ALGOL 60 and Church's Lambda-Notation: Part I.* Communications of the ACM. Feb, 1965.

- Extension to SECD
- Necessary to model assignment capability in Algol 60

J Operator

"We introduce into the SECD-machine an operation denoted by 'J', that is applicable to functions, or more precisely to *closures*, and modifies their subsequent exit behavior"

Peter J. Landin. *A Correspondence Between ALGOL 60 and Church's Lambda-Notation: Part I*. Communications of the ACM. Feb, 1965.

- Necessary to model *goto* in Algol 60

fin