Loudon
3 Design (1)

Language Design

- criteria :- efficiency ~~of execution.~~
  - readability
  - writeability
  - generality &orthogonality

- Efficiency
  - = of execution :
    - critical apps :- servers/OS.
      - computation intensive
        - sci. apps
        - movies.
    - optimization .
      - static typing
      - alias analysis
  - = of translation
    - one pass compilation ?
    - pascal & C
    - not ML - type analysis
    - dangling ptrs
  - = reliability
    - no unchecked ptrs or ~~[illegible]~~ indexes
  - = implementability
    - can write translator ?
    - type inference needs unification
    - type checking - graph algs.
    - Ada : types both inh & syn

JIT

  - = programmer effic.
  - = expressiveness
    - complex processes .
  - = maintainabilty

Loudon
3 Design (2)

3.3 <u>Regularity</u>
- principle of least surprise
- generality. - no special cases
- orthogonal
- uniformity.

• <u>Generality</u>
lack of : pascal - procs as parms but not vars.
(C simul via ptrs)
C - no nested fns.
arrays - no variable len (strings).
equality == in C identity but not similarity
== C++ strings.
can't extend (+) in Java.

• <u>Orthogonality</u>
. things behave diff in diff places?
. can't return arrays in C
. C: local vars only after {
. Java: prims passed by value; objs by ref.

• <u>Uniformity</u>
$\boxed{;}$ - end construct or separator.
C - req @ end of struct
proh @ fn

Algol68 - very complex due to ortho.

Loudon
3 Design (3)

Misc

- simplicity
  - Pascal for teaching
  - C for compiling on small sys.
  - ~~or straightjacket~~
  - straitjacket or no control
  - Lisp & prolog – simple compile
    – complex runtime
  - Basic simple – but hard to use.

  Einstein: "everything should be as simple as possible, but no simpler"

- Expressiveness
  - Lisp → recursion ; data = pgm.
  - Algol60 → structured prog.
  - OOP – more expressive than procedural
  - fn prog also.
  - Whorf's law
  - but obscure? $\boxed{\texttt{while (*p++=*Q++);}}$

- Extensibility
  - * add new features.
  - define new datatypes
    - " fns in a lib
    - " primitives
  - macro language
  - overloading & overriding
  - new opers? → $\boxed{\texttt{infixr 6 +++}}$ ← ML Haskell
  - math.

+

+.

Louden
3 Design (4)

- Restrictability
  - can newbie use a subset?
    ~~qui~~ to prog (12A text) — to the lang (tutorial)
  - subset language (PL/1 - PL/C, Fortran - WATFIV)
  - ex: concurrency & excns.
  - Perl : TMTOWTDT.
  - ~~#define for(*~~
  - #define while(X) for(;X;)

- Consistency
  - accepted notations
  - ops + - / from math, * from ?? but now, std.
  - prog, fn, var, white space
  - <u>law of least astonishment</u>

- Preciseness
  - more predictable xlators
  - less chance for optimiz
  - langu man. or ANSI/ISO std.

- M/C independence
  - cycle of lang design $\rightleftarrows$ m/c design
  - can it be free: - 32 bit 2's compl won. but not 20yrs ago
    - IEEE 754 flpt.

$$\begin{tikzcd} & \text{compiler} \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\ \text{machine} \arrow[rr, leftrightarrow] & & \text{lang} \end{tikzcd}$$

  - C many named consts for sizes.
  - sizeof(long) ≡? sizeof(ptr)

- Security
  - finger worm buffer overflow
  - type checking, bounds, ptr checking.
  - maximize # errors prohibited
  - Java - JNI
  - C++: y = static_cast<Foo>(x);

C Qsort

Loudon
3 Design (5)

C++ (Stroustrup)

- Simula 67 = 1st O.O.
- need fast compile/link
= "C with classes"

First (1980) Cfront : $C{+}{+} \rightarrow C$

- C compatibility
- incremental changes
- avoid "neat" (cute?) features
- what you don't use, ~~go~~ does not cost you.
- multiparadigm lang.
- subset learning.

standardization

- language + STL
- templates added recently
- RTTI

mistake: no std lib @ outset