

Language Design

Lowdon
3 Design (1)

- 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 ~~indexes~~ indexes

= implementability

- can write translator?

- type inference needs unification

- type checking - graph algs.

- Ada: types both inh & sym

= programmer effc.

= expressiveness

- complex processes.

= maintainability

JIT

3.3 Regularity

- principle of least surprise
- generality. - no special cases
- orthogonal
- uniformity.

• Generality

lack of : pascal - procs as params 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.

• Orthogonality

• 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.

• Uniformity

③ - end construct or separator.

C - req @ end of struct
proh @ fn

Algol68 - very complex due to ortho.

Misc

• simplicity

- Pascal for teaching
- C for computing on small sys.
- ~~- or straight j ket~~
- 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?

while (*p++ == *q++);

• Extensibility

* add new features.

- define new data types
 - " fns in a lib
 - " primitives
- macro language
- overloading & overriding
- new ops? →

infixr 6 ++

 ← ML Haskell
- math.

+

+.

• Restrictability

- can newbie use a subset?
- ~~Q~~ to prog (12A text) to the lang (tutorial)
- subset language (PL/I - PL/C
Fortran - WATFIV)
- ex: concurrency & exns.
- Perl: TM TOWTDT.
- ~~#define~~ ~~for(x)~~
- #define while(x) for(;x;)

• Consistency

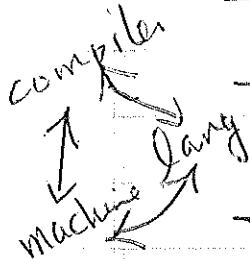
- accepted notations
- ops + - / from math * from ??
but now, std.
- prog, fn, var, white space
- law of least astonishment

• Preciseness

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

• M/C independence

- cycle of lang design \rightarrow m/c design



- can it be free: - 32 bit 2's compl won.
but not 20yrs ago
- IEEE 754 flpt.
- C many named const for sizes.
- sizeof(long) \equiv ? sizeof(ptr)

• Security

- finger worm buffer overflow
- type checking, bounds, ptr checking.
- maximize # errors prohibited
- Java - JNI
- C++: $y = \text{static_cast} \langle \text{Foo} \rangle (x);$

C
Qsort

London
3 Design (5)

C++ (Stroustrup)

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

Fvst (1980) Cfront : C++ → C

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

Standardization

- language + STL
- templates added recently
- RTTI

Mistake: no std lib @ outset

