Language design

Language design

- Key concepts:
  - Syntax, Semantics, Pragmatics
  - Language categories
  - Specification
  - Design concepts
  - Implementation
  - Standardization
  - Programming language evolution

Syntax, Semantic, Pragmatics

- Syntax
  - The correct grammar of the language
- Semantic
  - The meaning of a syntactically correct phrase
- Pragmatics
  - How to use the given phrase for a useful purpose

Syntax

- Help the lexer/parser
  - C declaration syntax: double (*funptr)(double);
- Help the programmer to write correct code
  - Pascal or C type of use ;
- Too lazy
  - PL/I generated missing **end** keyword
  - Algol68 begin (interchangable
  - Newline appears in strings?
- Too strict
  - Algol68 only implementation for **skip** statement

Syntax

- Goto?
- Exceptions?
- Block statement:
  - The **goto fail** error

```
if ( cond )
    goto fail;
    goto fail;
 i = 6;
```

Syntax

- (Dangling) **else** statement:

```
if ( i < 10 )
    if ( j < 20 ) i += j;
else
    i = j;
```

- Switch statement in C
  - 90% of case statements require break

Syntax

- C++11

```
vector<vector<Node>> parents;
int n = index_of_parent();
/// ...
for ( Node n in parents[n] )
```

Semantics

- The meaning of the code
  - Axiomatic

$$\{\ \mathtt{x}{=}\mathtt{n} \wedge \mathtt{y}{=}\mathtt{m}\ \}\ \mathtt{z}{:=}\mathtt{x};\ \mathtt{x}{:=}\mathtt{y};\ \mathtt{y}{:=}\mathtt{z}\ \{\ \mathtt{y}{=}\mathtt{n} \wedge \mathtt{x}{=}\mathtt{m}\ \}$$

  - Denotational

$$\mathcal{S}[\![\mathtt{z}{:=}\mathtt{x};\ \mathtt{x}{:=}\mathtt{y};\ \mathtt{y}{:=}\mathtt{z}]\!] = \mathcal{S}[\![\mathtt{y}{:=}\mathtt{z}]\!] \circ \mathcal{S}[\![\mathtt{x}{:=}\mathtt{y}]\!] \circ \mathcal{S}[\![\mathtt{z}{:=}\mathtt{x}]\!]$$

  - Operational

$$\frac{\langle \mathtt{z}{:=}\mathtt{x},\ s_0\rangle \to s_1 \qquad \langle \mathtt{x}{:=}\mathtt{y},\ s_1\rangle \to s_2}{\langle \mathtt{z}{:=}\mathtt{x};\ \mathtt{x}{:=}\mathtt{y},\ s_0\rangle \to s_2 \qquad \langle \mathtt{y}{:=}\mathtt{z},\ s_2\rangle \to s_3}$$

$$\langle \mathtt{z}{:=}\mathtt{x};\ \mathtt{x}{:=}\mathtt{y};\ \mathtt{y}{:=}\mathtt{z},\ s_0\rangle \to s_3$$

  - Textual

Pragmatics

- How to write good code
- Programmers practices, design rules
-
- Scala
  - Optimize for immutable
- C++
  - Use RAII, Pimpl, use const correctness
- C

  if (5 == strlen(str))

Language specification

- Fortran: BNF
- Pascal: EBNF, „Railways notation”
- ALGOL68: first textual
  - After 1973 revised in Van Wijngaarden grammar
  - Context sensitive
  - Turing complete
- C/C++ textual + abstract machine

Language design concepts

- Well-defined syntax and semantics
- Expressivity
  - In APL 256 operators
  - Redundancy is important
- Orthogonality
  - C++: protected abstract virtual base pure virtual private destructor

Tom Cargill

- If you think C++ is not overly complicated, just what is a protected abstract virtual base pure virtual private destructor, and when was the last time you needed one? — Tom Cargill, C++ Journal, Fall 1990

- The first 90% of the code accounts for the first 90% of the development time. The remaining 10% of the code accounts for the other 90% of the development time.

Language design concepts

- Generality
  - C++ templates
  - Java generics?
- Modularity
  - Java package vs. C++ namespace
  - C++ included headers?, Erlang flat modules?

Language design concepts

- Portability
  - Source/Bytecode/Binary
  - Pascal P code, COBOL
- Performance
  - Garbage collection?
  - Optimizations vs. Debugging
- Learnability

Implementation

- Compilation
  - Phases: (Preprocessing), Compiling, Linking
  - Static or dynamic linking
  - Generates HW and OS-specific executable
  - Effective optimizations
- Interpretation
  - Faster developing process
  - Less correctness-checking possibilities

Implementation

- Hybrid model
  - Compiler generates platform independent intermediate code
  - Intermediate code executed by “virtual machine”
  - Fair correctness checking and optimization
  - More optimization: Just-in-time compilers
- Samples
  - Pascal P-code, Java virtual machine, MS IL

Standardization

- Reasons
  - Portability of source
  - Maintanability
  - Portability of programmers
  - Acceptability
  - Faster development
- Standard library must included
- C++ ISO (since 1998)
- C# ECMA-334 C# version 2.0
- Java nope

Language evolution

- New features/keywords
  - Reverse compatibility issues
  - Depricated elements
  - Silent semantic changes?
- Successful
  - C++: delete functions, auto keyword, overload
- Issues
  - Python 2 → Python 3
  - C to C++