Motivation

- Explore different abstractions available in different languages.
- Allows choosing language appropriate to problem.
- Better understanding of languages makes it easier to learn new languages.
- Become an overall better programmer.
- Simulate features in languages which lack them.
- Facilitate better use of language technology in different situations. For example, in analyzing structured files or designing extension/configuration languages.

Motivation Continued

- Understand relatively obscure features like unions in C/C++, multiple-inheritance in C++ and CLOS, multi-dispatching in CLOS, varargs in C/C++.
- Choose among alternatives. In C++ use copy constructors to minimize cost of construction. With poor compilers, choice between `x*x` versus `x^2`, arrays versus pointers in C.
- Better understanding aids use of tools like debuggers, assemblers and linkers.

Brief History

Plankalkul: Konrad Zuse, 1945. Surprisingly advanced, considering its chronology. Had Eiffel-style assertions. 2-dimensional notation. Not widely known. See the Wikipedia article.

Fortran: Backus, 1954. Initially for IBM 704. Emphasis on *formula translation* of mathematical formula to machine language. Efficiency of execution very important. Initially had primitive control structures: a indexed `DO` loop; a 3-way arithmetic `IF`. Uses a fixed-column format with spaces ignored. Highly successful and still extremely popular. Latest version is Fortran 2003.

Brief History Continued

Lisp: McCarthy. 1960. Used for AI and symbolic processing. Only 2 basic data types: *atoms* and *pairs*. Functional programming language with imperative features. Supported recursion. Simple and regular syntax. Garbage collection. Programs and data have same syntax: parenthesized postfix. Initially interpreted. Possible to change program at runtime. Highly successful. Current versions include Common Lisp and Scheme.

Cobol: DoD Committee, 1960. Used for business applications. Easy to produce reports. Hierarchical data structures. Verbose syntax.

Brief History Continued

Algol: International committee, 1960. Academic origins. Block structure, recursion, call by value and call by name, local dynamic arrays. Never popular as a programming language but used for communicating algorithms. Highly influential through its successors Pascal, C, ...

APL: Iverson, 1960. Array-operators. Write-only programs. Reasonably popular for scientific and engineering applications.

PL/I: IBM, 1965. Supposed to be a universal language. Combination of features from Fortran, Algol and Cobol. A very complex language. Not terribly successful.

Snobol: Griswold, 196x. Text processing. Descendent Icon.

Brief History Continued

Simula: Nygaard and Dahl, 1967. Object-oriented features.

Pascal: Wirth, 1971. Simplified Algol. User-defined data types, records. Almost too simple for practical use without changes. Widely used for instruction. Portable implementation using a P-code virtual machine.

C: Kernighan and Ritchie, 1972. Emphasis on systems programming. Low level language (called a *portable assembly language*) which has evolved to support fairly high-level features. Highly successful.

Brief History Continued

Prolog: Roussel, 1975. Programming in Logic. Origins in automated theorem proving. Uses a general form of pattern matching for data-structure access and construction.

Smalltalk: Kay, 1980. Object-oriented language. Very rich set of libraries and an integrated programming environment. Relatively widely used at the time. Extremely influential.

Brief History Continued

C++: Stroustrup, 1984. C with added OO features. Has evolved into a extremely complex language. Widely used.

Java: Gosling, 1995. A simpler version of C++. Got a boost from WWW browsers supporting Java *applets* (which never took off). Very popular currently.

C♯: Microsoft, 2002. A larger version of Java tied to MS `.NET` platform. Relatively popular due to MS backing.

Brief History Continued

Scripting languages:

Perl: Larry Wall, 1987. Became very popular with the rise of the WWW in the 1990's. Weak OO. Perl 5 still widely used but fading. Perl 6 (a new language) was introduced after over a decade of development.

Python: Guido Van Rossum, around 1990. OO scripting language. Indentation part of syntax. Currently most popular scripting language.

Ruby: Yukihiro "Matz" Matsumoto, around 1995. OO scripting language. Minimal syntax, ideal as internal domain-specific language. Popularized by Ruby-On-Rails web framework. Meta-programming.

Brief History Continued

Recent languages:

Go: Google, 2007. Statically-typed with implicit typing, scalable language, in the tradition of C.
Duck-typed interfaces instead of inheritance.
Light-weight concurrency.

Swift: Apple, 2014. Replacement for Objective-C.
Rumors of Google using it to replace Java.

Universal Client Language

As browsers become a universal client, the programming language used in most browsers Javascript is becoming the universal client language.

- Underrated language because of poor browser DOM implementations.
- Resurgence because of AJAX client applications.
- Borrows concepts from Scheme and Self.
- Prototype-based object model without classes.
- Recent trends for use on server side (nodejs) ... one language on both client and server?
- Newer languages like coffeescript, typescript, elm implemented by compiling into javascript.

Universal Target Architectures

- Number of general purpose hardware architectures are shrinking: x86, ARM, SPARC, Power-PC, etc.
- Previously new languages were often compiled to C.
- Current new languages target the JVM (Scala, JRuby,Jython, Clojure) or the CLR (F#, Iron Ruby, Iron Python).
- Java 7 JVM has special support for dynamic languages with addition of `invokedynamic`.