A brief history of programming languages

Points

- Lots! (Read the textbook, Chapter 2. ☺)

Read the textbook for a thorough treatment of this fascinating subject. In class we will only discuss the milestones.

Pioneers of programming

Babbage: the Analytical Engine.

Zuse: Plankalkül, a notation, never implemented, that could have been the basis of many existing languages.

Very low-level languages

They were (and still are) machine-dependent coding systems: initially binary, then symbolic machine languages—one, or maybe one class, per computer model.

Fortran

The first effectively implemented high-level language that introduced variables as we know them now, procedures, statement labels and much more. It had unique (not always progressive) features. Still widely used; the features of the newest version, Fortran 90, have converged toward other languages.

Algol 60

It was the first to have block structure, recursion, and a formal definition. It is unused, but it is the ancestor of most contemporary languages.

As far as programming language design goes, Algol 60 was undoubtedly the most important innovation in history (so far ☺).

Cobol

Business-oriented computations, a very strict program organization, poor control structures, elaborate data structures, record type introduced for the first time. Very popular in business and government, much less at universities. Revived for a while with the Y2K scare—why?

PL/I

A mix of Fortran, Algol 60, Cobol—an attempt at all-out generality. Pushed by IBM. Not used much today. Interesting feature: event handling.

Basic

The first language of personal computing, the first language of many programmers. Very simple, limited. Present-day extensions are full-fledged languages—not easy to learn any more.

Simula 67

An extension of Algol 60 for simulation of concurrent processes. Introduced the concept of classes and encapsulation; forerunner of Smalltalk and C++. Now unused.

Algol 68

A very elegant design, full orthogonality—extremely difficult to implement. A clever formal description, unfortunately hard to grasp by most users.

Pascal

A conceptually simplified and cleaned-up successor of Algol 60. A language for teaching structured programming, and an excellent first language of instruction. Its later extensions are full-fledged systems programming packages.

Modula-2

A better, conceptually uniform Pascal, with mechanisms to program concurrency (many processes running in parallel). Not used as much as it deserves. Its successors (Modula-3 and Oberon) are even more conceptually appealing, practically useful—and not popular at all.

Ada

The result of an elaborate design process, and a more successful attempt at generality. Completely standard: there are no dialects. There are, however, two standards: Ada 83 (the original), and Ada 95.

Ada has been designed to support concurrency in a very neat, systematic way.

C

The implementation language of Unix, a tool for systems programming and a software development language of choice for personal computers. Once fashionable. Dangerous if not used properly, not recommended to novice programmers. Low-level.

Lisp

Historically, one of the first languages, based on the concept of computing by evaluating functions. Good for symbolic computing, for years the only language for Artificial Intelligence work. Many dialects, two standards (Scheme, Common Lisp). Nice programming environments. Its successors are very elegant (Miranda, ML, Haskell) but not nearly as popular—and even Lisp has only a limited following.

Prolog

A very high-level language based on a subset of logic as a programming formalism. Very powerful. Declarative, non-deterministic (built-in backtracking). Associative memory, pattern-directed procedure invocation. In skilled hands it is a very strong tool.

Smalltalk

It is the purest object-oriented language yet. Comes complete with a graphical interface and an integrated programming environment.

C++

An object-oriented extension of the imperative language C. Impure hybrid, but very fashionable.

Java

A neat, cleaned up, sized-down reworking of C++, pure object-orientation (though not as consistent as Smalltalk), designed for Internet programming, said (incorrectly) to be slow. New fashion.

Trends: merging programming paradigms.

- Object-oriented extensions: C++, dialects of Lisp (CLOS), variants of Prolog (Prolog++).
- Logic programming combined with functional programming (very clever, but only experimental).
- Beyond languages: CASE tools, once very popular in large-scale software engineering.

Most languages are sequential: one processor, one process. Ada is a language designed to support concurrency: many processes running in parallel.

Summary

....
....
....
....
....
....
....
....
....
....
....
....
....
....
....
....