UNIVERSITÄT
BERN

Programming Languages

Prof. O. Nierstrasz

Spring Semester 2014

What is a Programming Language?

> *A programming language is a notational system for describing computation in a machine-readable and human-readable form.*
>
> *— Louden*

- A formal language for describing computation?
- A “user interface” to a computer?
- Syntax + semantics?
- Compiler, or interpreter, or translator?
- A tool to support a programming model?

What is a Programming Language? (II)

A programming language is a tool for developing executable models for a class of problem domains.

Paradigms (Model)

How do different language paradigms support problem-solving?

Foundations

What are the foundations of programming languages?

Semantics

How can we understand the semantics of programming languages?

Generations of Programming Languages

1GL: machine codes

2GL: symbolic assemblers

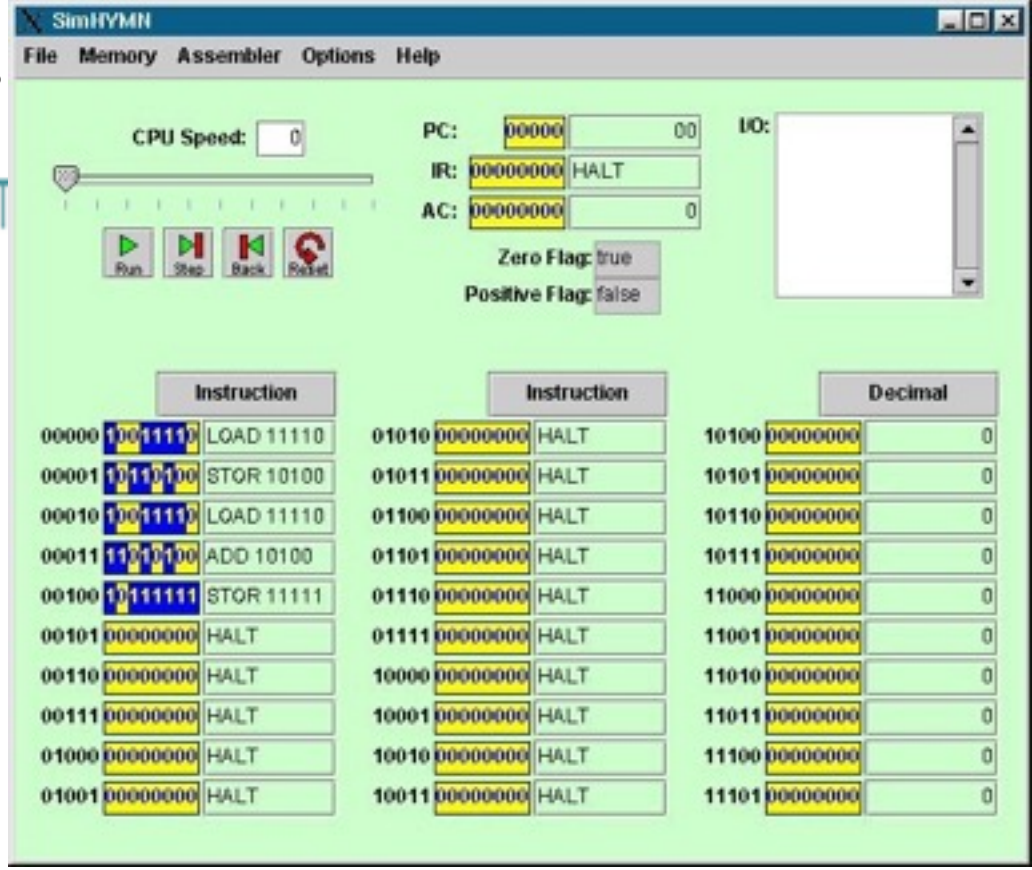
3GL: (machine-independent) imperative languages (FORTRAN, Pascal, C ...)

4GL: domain specific application generators

5GL: AI languages …

Each generation is at a higher level of abstraction

First Generation Languages (Machine Language)



- Programming computers using the CPU's instruction set
- Also known as **Machine Language**

Machine Code File

A software file which contains the instructions from the CPU's instruction set.

First Generation Languages (Machine Language)

Advantages of First Gen.

- Software programs execute (run) relatively very quickly
- Software programs are relatively small in size
- (Insignificant advantages today)

Disadvantages of First Gen.

- Difficult to write, very detailed and takes a long time
- Difficult to read
- Difficult to debug

debug = the process to find mistakes in a software program

Second Generation Languages (Assembly Language)

Instruction Set	Instruction
0001	Move
0010	Compare
0011	Bit test
0100	Bit clear
0101	Bit set
0110	Add
0111	See group 10
1000	See groups 11, 13, 14
1001	Move byte

```
INPORT:     EQU   20h          ;input and output ports are defined by names
OUTPORT:    EQU   21h
            ORG   0000h        ;the program is forced to begin at 0100h in order to avoid
            JP    START        ;conflicts with the "interrupt" zone
            ORG   0100h
START:      LD    SP,3FFFh     ;Stack Pointer is initialized, and A is given a default startup
            LD    A,00h        ;value (zero)
            OUT   (OUTPORT),A
STATE_A:    CALL  WAITL        ;CLOCK control subroutine
            LD    A,0Bh        ;0Bh=0000.1011b:S0,LOAD and CLEAR are active
            OUT   (OUTPORT),A
STATE_B:    CALL  WAITL
            LD    A,08h        ;08h=0000.1000b:only S0 is active (shift right is being
            OUT   (OUTPORT),A  ;performed)
            IN    A,(INPORT)
            BIT   2,A          ;test on the IN bit:if low,back to reset state (a);else,
            JP    Z,STATE_A    ;ahead to state (c) (valid string incoming)
```

Assembly Language = The English-like instructions which are equivalent to the CPU's instruction set

Source Code= The actual instructions written by a programmer

Compiler = Software which translates source code instructions of a particular language into machine code

Second Generation Languages (Assembly Language)

Question: Which of these two files (source code file or machine code file) will the user need to run this software program?

Advantages of Second Gen.

- Easier to read than first gen.
- Easier to write than first gen.
- Easier to debug than first gen.

Disadvantages of Second Gen.

- Still very difficult to write programs

Third Generation Languages
(High level languages)

Languages which are somewhere between machine language and the human language.

FORTRAN (Formula Translation) - 1950's

Language to allow scientists and engineers to program computers.

COBOL (Common Business Oriented Language) - 1960

Language primarily designed for US government and defense contractors to program business applications on the computer. Grace Hopper was one of the developers of COBOL.

BASIC (Beginner's All-purpose Symbolic Code) - 1960's

Alternative language to FORTRAN for beginning programming students.

Third Generation Languages
(High level languages)

Pascal (named after Blaise Pascal, 17th century French mathematician) - 1970's

Language to teach proper structured programming.

> **Structured programming** = Programming technique used to make programming more productive and easier to write. Stresses simplistic, modular programs.

ADA (named after Ada Lovelace (programmed the 19th century 'analytical engine') - late 1970's

Language developed to replace COBOL.

Third Generation Languages
(High level languages)

C (successor to BCPL or "B") - 1970's

Popular programming language on computers from microcomputers to super computers.

Faster and more efficient language. Very powerful language.

Source code example of a C Program (Displays *Hello World!* on the screen.)

```
#include <stdio.h>
main()
{
    printf("Hello World!");
}
```

C++ (pronounced "C plus plus") - 1980's

Object oriented language which is compatible with C.

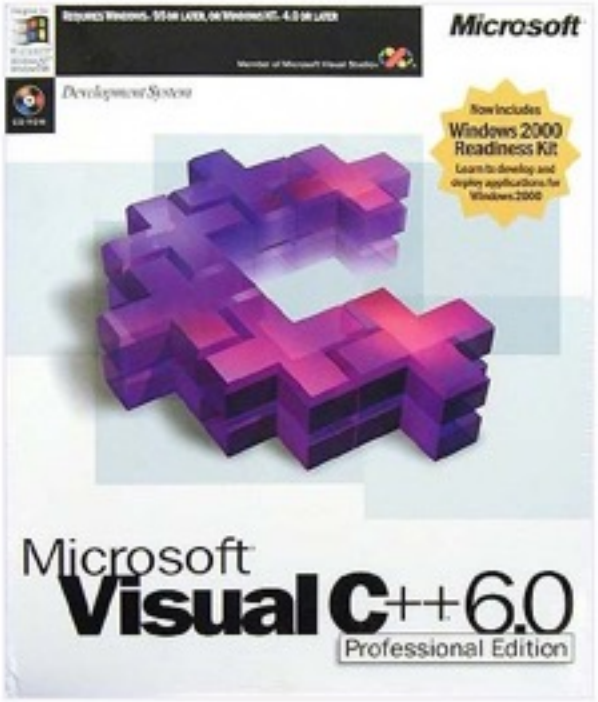
Third Generation Languages (High level languages)

Advantages

- Easier to read, write and debug
- Faster creation of programs

Disadvantages

- Still not a tool for the average user to create software programs
- Requires very good knowledge of programming and of the language



Third Generation Languages
(High level languages)

Advantages

- Easier to read, write and debug
- Faster creation of programs

Disadvantages

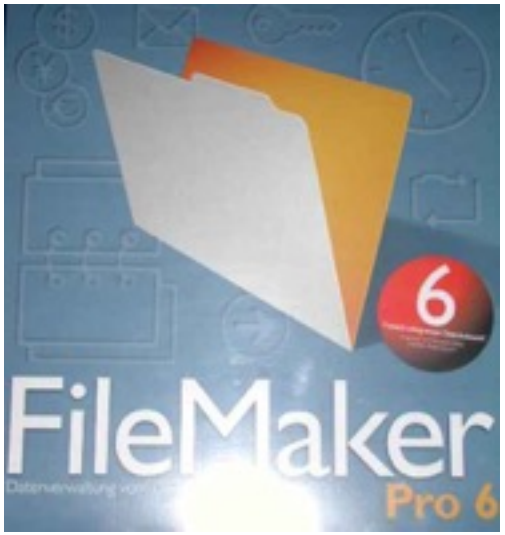
- Still not a tool for the average user to create software programs
- Requires very good knowledge of programming and of the language

Fourth Generation Languages

Languages which are more like natural human languages

- uses English phrases
- common with data base languages

search for name equals “graziani” and state equals “ca”

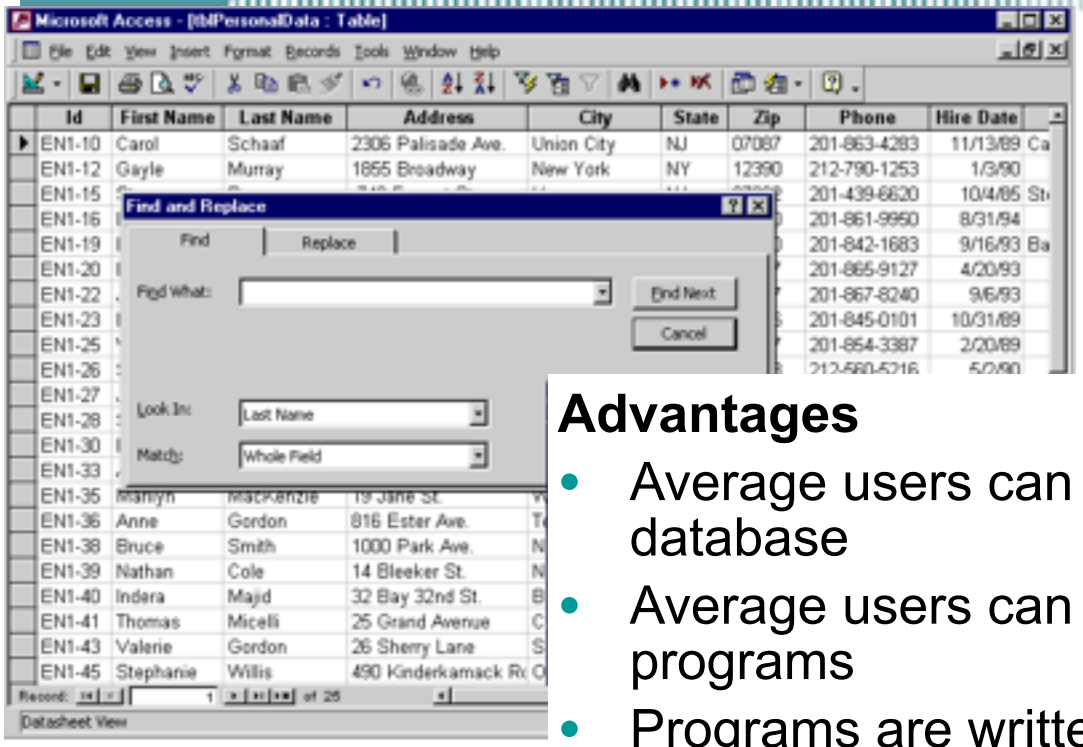


Examples

dBase FoxPro Access

Oracle Informix SQL

Fourth Generation Languages



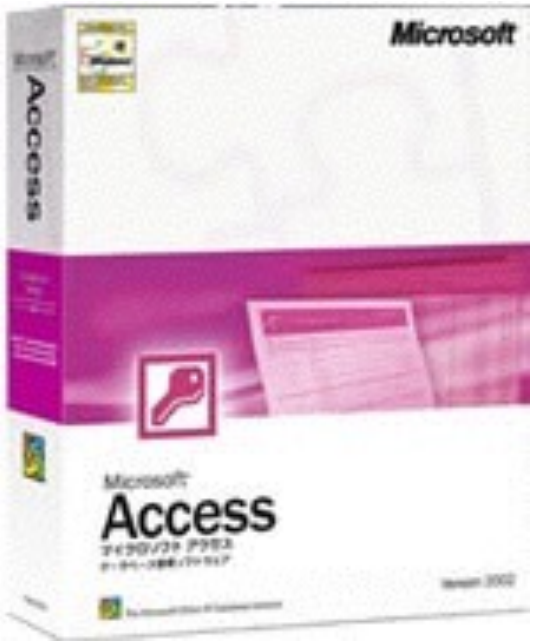
Advantages

- Average users can quickly learn to “query” the database
- Average users can easily learn how to write programs
- Programs are written faster

Disadvantages

- Can not write sophisticate programs like word processors, graphics, etc.
- Mostly for data base applications like phone information.

Databases and Relationships



How do Programming Languages Differ?

Common Constructs:

> basic data types (numbers, etc.); variables; expressions; statements; keywords; control constructs; procedures; comments; errors ...

Uncommon Constructs:

> type declarations; special types (strings, arrays, matrices, ...); sequential execution; concurrency constructs; packages/modules; objects; general functions; generics; modifiable state; ...

Programming Paradigms

A programming language is a problem-solving tool.

Traditional style:	program = algorithms + data *good for decomposition*
Functional style:	program = functions ∘ functions *good for searching*
Logic programming style:	program = facts + rules *good for reasoning*
Object-oriented style:	program = objects + messages *good for modeling*

A Brief Chronology

Early 1950s		*“order codes” (primitive assemblers)*
1957	FORTRAN	*the first high-level programming language*
1958	ALGOL	*the first modern, imperative language*
1960	LISP, COBOL	*Interactive programming; business programming*
1962	APL, SIMULA	*the birth of OOP (SIMULA)*
1964	BASIC, PL/I	
1966	ISWIM	*first modern functional language (a proposal)*
1970	Prolog	*logic programming is born*
1972	C	*the systems programming language*
1975	Pascal, Scheme	*two teaching languages*
1978	CSP	*Concurrency*
1978	FP	*Backus’ proposal*
1983	Smalltalk-80, Ada	*OOP is reinvented*
1984	Standard ML	*FP becomes mainstream (?)*
1986	C++, Eiffel	*OOP is reinvented (again)*

Fortran

History

- *John Backus (1953) write programs in mathematical notation, and generate code comparable to good assembly programs.*
- No language design
- Most effort spent on code generation and optimization
- FORTRAN I released April 1957; working by April 1958
- The current standard is FORTRAN 2018
- https://en.wikipedia.org/wiki/Fortran - History

Fortran ...

Innovations

- Notation for functions
- Assignments to variables of complex expressions
- DO loops
- Comments
- Input/output formats
- Machine-independence

Successes

- Easy to learn; high level
- Promoted by IBM; addressed large user base
- (scientific computing)

“Hello World” in FORTRAN

```
   PROGRAM HELLO
   DO 10, I=1,10
   PRINT *,'Hello World'
10 CONTINUE
   STOP
   END
```

All examples from the ACM "Hello World" project:
www2.latech.edu/~acm/HelloWorld.shtml

ALGOL 60

History

- *Committee of PL experts formed in 1955 to design universal, machine-independent, algorithmic language*
- First version (ALGOL 58) never implemented

Innovations

- BNF (Backus-Naur Form) introduced to define syntax (led to syntax-directed compilers)
- First block-structured language; variables with local scope
- Structured control statements
- Recursive procedures
- Variable size arrays

Successes

- Highly influenced design of other PLs but never displaced FORTRAN

“Hello World” in BEALGOL

```
BEGIN
FILE F (KIND=REMOTE);
EBCDIC ARRAY E [0:11];
REPLACE E BY "HELLO WORLD!";
WHILE TRUE DO
   BEGIN
   WRITE (F, *, E);
   END;
END.
```

COBOL

History

> *Designed by committee of US computer manufacturers*
> Targeted business applications
> Intended to be readable by managers (!)

Innovations

> Separate descriptions of environment, data, and processes

Successes

> Adopted as de facto standard by US DOD
> Stable standard for 25 years
> Still *the most widely used PL* for business applications (!)
> COBOL 2014 - History https://en.wikipedia.org/wiki/COBOL

“Hello World” in COBOL

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID.     HELLOWORLD.
000300 DATE-WRITTEN.   02/05/96        21:04.
000400* AUTHOR BRIAN COLLINS
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
001000 DATA DIVISION.
001100 FILE SECTION.
100000 PROCEDURE DIVISION.
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400     DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100500     DISPLAY "HELLO, WORLD." LINE 15 POSITION 10.
100600     STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800     EXIT.
```

PL/1

History

- *Designed by committee of IBM and users (early 1960s)*
- Intended as (large) general-purpose language for broad classes of applications

Innovations

- Support for concurrency
- Exception-handling on conditions

Successes

- Achieved both run-time efficiency and flexibility
- First “complete” general purpose language

“Hello World” in PL/1

```
HELLO:    PROCEDURE OPTIONS (MAIN);

          /* A PROGRAM TO OUTPUT HELLO WORLD */
          FLAG = 0;

LOOP:     DO WHILE (FLAG = 0);
             PUT SKIP DATA('HELLO WORLD!');
          END LOOP;

END HELLO;
```

“Hello World” in Functional Languages

SML - 1983

```
print("hello world!\n");
```

Standard ML (SML; "Standard Meta Language") is a general-purpose, modular, functional programming language with compile-time type checking and type inference.

It is popular among compiler writers and programming language researchers, as well as in the development of theorem provers.

“Hello World” in Functional Languages

Haskell - 1990

general-purpose compiled purely functional programming language

The latest standard of Haskell is Haskell 2010. As of May 2016, a group is working on the next version, Haskell 2020.[29]

```
hello() = print "Hello World"
```

Prolog

History

> Originated at U. Marseilles (early 1970s), and compilers developed at Marseilles and Edinburgh (mid to late 1970s)

Innovations

> Theorem proving paradigm

> Programs as sets of clauses: facts, rules and questions

Successes

> Prototypical logic programming language

> Used in Japanese Fifth Generation Initiative

“Hello World” in Prolog

```
hello :- printstring("HELLO WORLD!!!!").

printstring([]).
printstring([H|T]) :- put(H), printstring(T).
```

Object-Oriented Languages

History

- ***Simula*** was developed by Nygaard and Dahl (early 1960s) in Oslo as a language for simulation programming, by adding *classes and inheritance* to ALGOL 60

```
Begin
   while 1 = 1 do begin
      outtext ("Hello World!");
      outimage;
   end;
End;
```

- ***Smalltalk*** was developed by Xerox PARC (early 1970s)

```
Transcript show:'Hello World';cr
```

Object-Oriented Languages

Innovations

- *Encapsulation* of data and operations
- *Inheritance* to share behaviour and interfaces

Successes

- Smalltalk project pioneered OOP user interfaces
- Large commercial impact since mid 1980s
- New languages: C++, Objective C, Eiffel, Beta, Oberon, Self, Perl 5, Python, Java, Ada 95 ...

Interactive Languages

BASIC

- Developed at Dartmouth College in mid 1960s
- Minimal; easy to learn
- Incorporated basic O/S commands

```
10 print "Hello World!"
20 goto 10
```

...

Interactive Languages ...

APL

- Developed by Ken Iverson for short description of numerical algorithms
- Large, 52 characters in addition to alphanumerics)
- Objects are arrays (lists, tables or matrices)
- Operator-driven (power comes from composing array operators)
- No operator precedence (**statements parsed right to left**)

```
'HELLO WORLD'
```

Special-Purpose Languages

SNOBOL

- First successful string manipulation language
- Influenced design of text editors more than other PLs
- String operations: pattern-matching and substitution
- Arrays and associative arrays (tables)
- Variable-length strings

```
   OUTPUT = 'Hello World!'
END
```

...

Lisp

- Performs computations on symbolic expressions
- Symbolic expressions are represented as *lists*
- Small set of constructor/selector operations to create and manipulate lists
- Recursive rather than iterative control
- *No distinction between data and programs*
- First PL to implement storage management by garbage collection

```
(DEFUN HELLO-WORLD ()
       (PRINT (LIST 'HELLO 'WORLD)))
```

4GLs

"Problem-oriented" languages

> PLs for "non-programmers"
> Very High Level (VHL) languages for specific problem domains

Classes of 4GLs (no clear boundaries)

> Application generators
> Query languages
> Decision-support languages

Successes

> Highly popular

“Hello World” in RPG

```
H
FSCREEN O F 80 80                                          CRT
C                                EXCPT
OSCREEN E 1
O                                                 12 'HELLO WORLD!'
```

“Hello World” in SQL

```
CREATE TABLE HELLO (HELLO CHAR(12))
UPDATE HELLO
   SET HELLO = 'HELLO WORLD!'
SELECT * FROM HELLO
```

Scripting Languages

History

Countless “shell languages” and “command languages” for operating systems and configurable applications

- ***Unix shell*** (ca. 1971) developed as user shell and scripting tool

```
echo "Hello, World!"
```

- ***HyperTalk*** (1987) was developed at Apple to script HyperCard stacks

```
on OpenStack
    show message box
    put "Hello World!" into message box
end OpenStack
```

- ***TCL*** (1990) developed as embedding language and scripting language for X windows applications (via Tk)

```
puts "Hello World "
```

- ***Perl*** (~1990) became de facto web scripting language

```
print "Hello, World!\n";
```

Scripting Languages ...

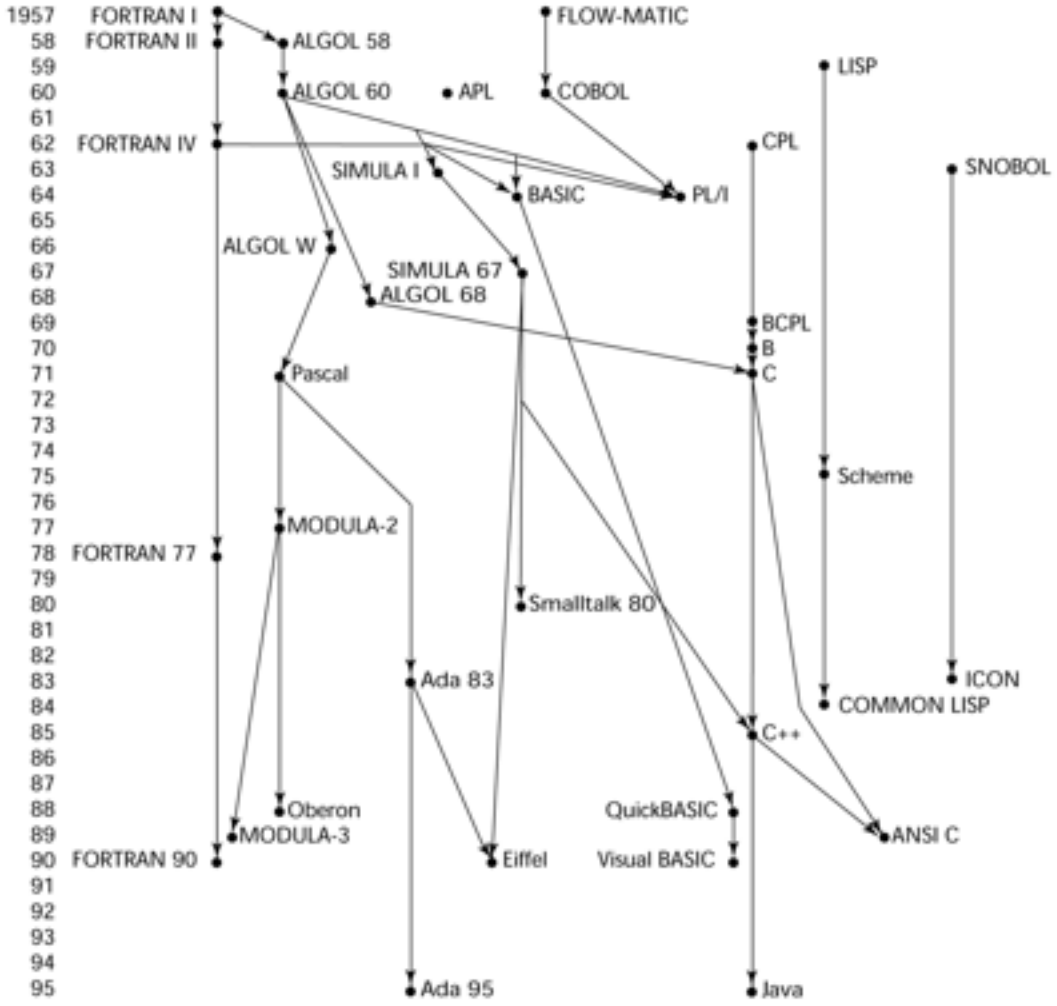
Innovations

- Pipes and filters (Unix shell)
- Generalized embedding/command languages (TCL)

Successes

- Unix Shell, awk, emacs, HyperTalk, AppleTalk, TCL, Python, Perl, VisualBasic ...

Genealogy of Common Languages



The future?

- Dynamic languages
  - very active
- Domain-specific languages
  - very active
- Visual languages
  - many developments
- Modeling languages
  - Combine models (UML,…)