Programming Languages

Introduction

1

Overview

- **Motivation**
- **Why study programming languages?**
- **Some key concepts**

2

What is a programming language?

3
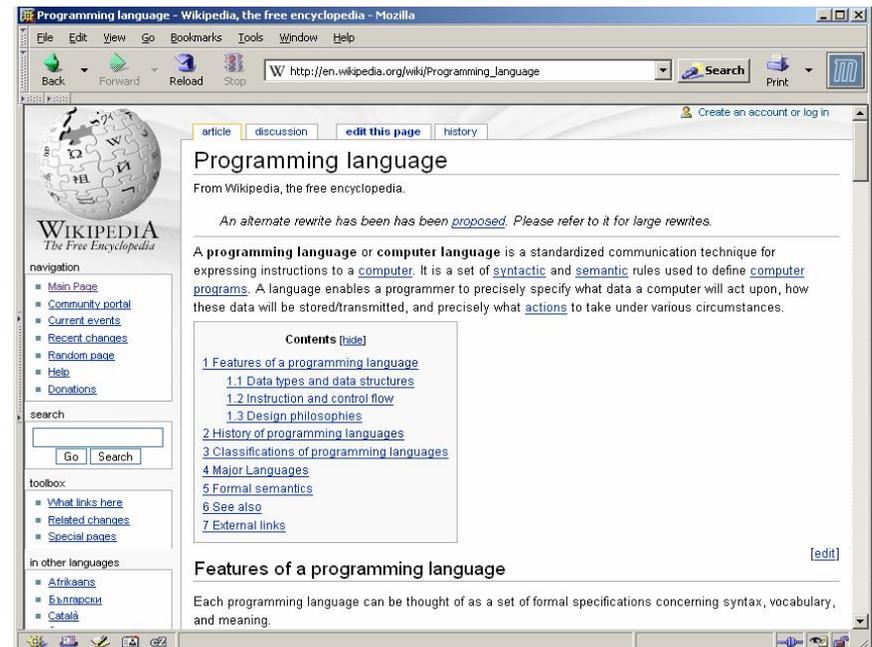
What is a programming language?

"...there is no agreement on what a programming language really is and what its main purpose is supposed to be. Is a programming language a tool for instructing machines? A means of communicating between programmers? A vehicle for expressing high-level designs? A notation for algorithms? A way of expressing relationships between concepts? A tool for experimentation? A means of controlling computerized devices? My view is that a general-purpose programming language must be all of those to serve its diverse set of users. The only thing a language cannot be – and survive – is a mere collection of ''neat'' features."

-- Bjarne Stroustrup, The Design and Evolution of C++

http://www.cs.umbc.edu/331/papers/dne_notes.pdf

5

On language and thought (1)

"The Language of Thought Hypothesis (LOTH) postulates that thought and thinking take place in a mental language. This language consists of a system of representations that is physically realized in the brain of thinkers and has a combinatorial syntax (and semantics) such that operations on representations are causally sensitive only to the syntactic properties of representations. …"

-- Stanford Encyclopedia of Philosophy

-- http://plato.stanford.edu/entries/language-thought/

6

On language and thought (2)

The tools we use have a profound (and devious!) influence on our thinking habits, and therefore, on our thinking abilities.

-- Edsger Dijkstra, *How do we tell truths that might hurt?,*

http://www.cs.umbc.edu/331/papers/ewd498.htm

Edsger Wybe Dijkstra (11 May 1930 -- 6 August 2002), http://www.cs.utexas.edu/users/EWD/

Professor Edsger Wybe Dijkstra, a noted pioneer of the science and industry of computing, died after a long struggle with cancer on 6 August 2002 at his home in Neunen, the Netherlands.

l

7

On languages and thought (3)
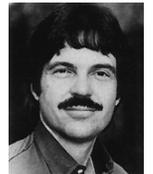
"What doesn't exist are really powerful general forms of arguing with computers right now. So we have to have special orders coming in on special cases and then think up ways to do it. Some of these are generalizable and eventually you will get an actual engineering discipline."

-- Alan Kay, Educom Review

Alan Kay is one of the inventors of the Smalltalk programming language and one of the fathers of the idea of OOP. He is the conceiver of the laptop computer and the architect of the modern windowing GUI.

8

Chapter One

Preliminaries, including

  –Why study PL concepts?
  –Programming domains
  –PL evaluation criteria
  –What influences PL design?
  –Tradeoffs faced by programming languages
  –Implementation methods
  –Programming environments

9

Why study Programming Language Concepts?

- Increased capacity to express programming concepts
- Improved background for choosing appropriate languages
- Increased ability to learn new languages
- Understanding the significance of implementation
- Increased ability to design new languages
- Mastering different programming paradigms

10

Programming Domains

- Scientific applications
- Business applications
- Artificial intelligence
- Systems programming
- Scripting languages
- Special purpose languages

11

Language Evaluation Criteria

- Readability
- Writability
- Reliability
- Cost
- Etc…

12

Evaluation Criteria: Readability

How is it for one to read and understand programs
   written in the PL?
Arguably the most important criterion!
Factors effecting readability include:
   – Overall simplicity
      » Too many features is bad as is a multiplicity
         of features
   – Orthogonality
      » Makes the language easy to learn and read
      » Meaning is context independent
   – Control statements
   – Data type and structures
   – Syntax considerations

13

Evaluation Criteria: Writability

How easy is it to write programs in the
   language?
Factors effecting writability:
   – Simplicity and orthogonality
   – Support for abstraction
   – Expressivity
   – Fit for the domain and problem

14

Evaluation Criteria: Reliability

Factors:

   - Type checking
   - Exception handling
   - Aliasing
   - Readability and writability

15

Evaluation Criteria: Cost

Categories:
   – Programmer training
   – Software creation
   – Compilation
   – Execution
   – Compiler cost
   – Poor reliability
   – Maintenance

16

Evaluation Criteria: others

Portability

Generality

Well-definedness

Etc…

17

Language Design Influences
Computer architecture

- We use imperative languages, at least in part, because we use **von Neumann** machines
  - John von Neuman is generally considered to be the inventor of the "stored program" machines - the class to which most of today's computers belong.
  - CPU+memory which contains both program and data
- Focus on moving data and program instructions between registers in CPU to memory locations

18

Von Neumann Architecture
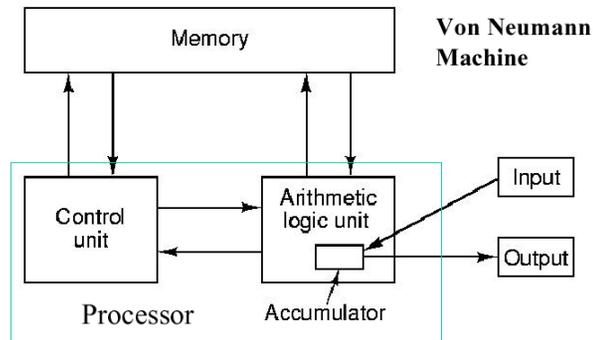
Language Design Influences:
Programming methodologies

- *50s and early 60s:* Simple applications; worry about machine efficiency
- *Late 60s:* People efficiency became important; readability, better control structures. maintainability
- *Late 70s:* Data abstraction
- *Middle 80s:* Object-oriented programming
- *90s:* distributed programs, internet, web
- 00s: Autonomic systems?, pervasive computing? Genetic programming? Semantic web?

19

20

Language Categories

The big four:

Imperative or procedural (e.g., Fortran, C)

Functional (e.g., Lisp, ML)

Rule based (e.g. Prolog, Jess)

Object-oriented (e.g. Smalltalk, Java)

Others:

Scripting (e.g., Perl, Tcl/Tk, PHP)

Constraint (e.g., Eclipse)

Concurrent (Occam)

…

21

Language Design Trade-offs

Reliability versus cost of execution

Ada, unlike C, checks all array indices to ensure proper range.

Writability versus readability

(2 = 0 +.= T o./ T) / T <- iN is an APL one liner that produces a list of the prime numbers from 1 to N inclusive.

Flexibility versus safety

C, unlike Java, allows one to do arithmetic on pointers.

22

Implementation methods

• Direct execution by hardware
  – E.g., machine language
• Compilation to another language
  – e.g., C compiles to machine language for an Intel Pentium 4
• Interpretation
  – Direct execution by software
  – E.g., csh, Lisp (traditionally)
• Hybrid
 – Compilation to another language (aka bytecode) which is then interpreted
 – e.g., Java, Perl
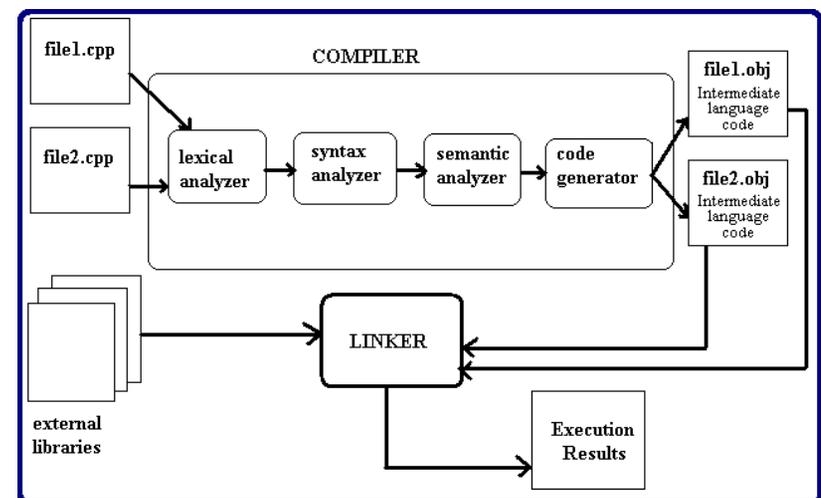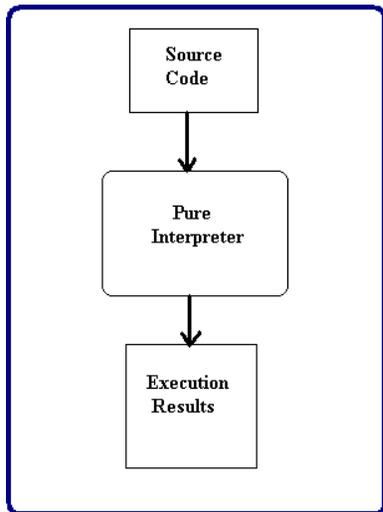
23

Compilation

24

Interpretation

25

Hybrid
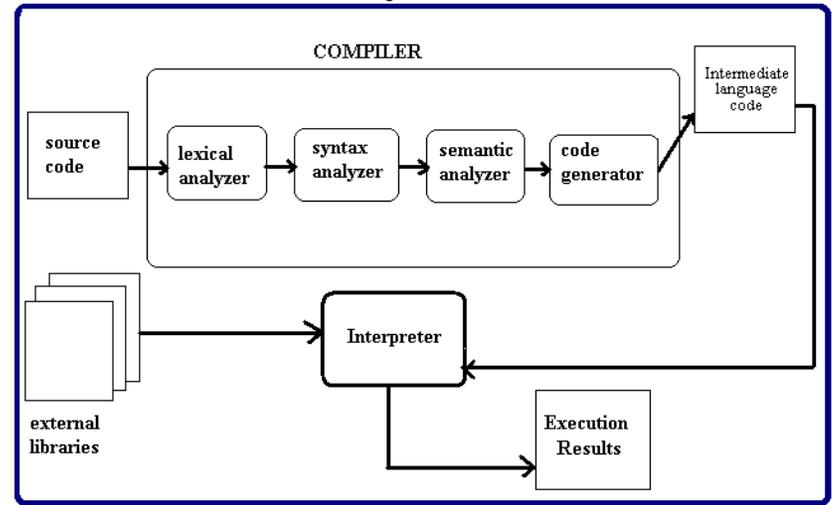
26

Implementation issues

1. Complexity of compiler/interpreter

2. Translation speed

3. Execution speed

4. Code portability

5. Code compactness

6. Debugging ease

compile   hybrid   interpret
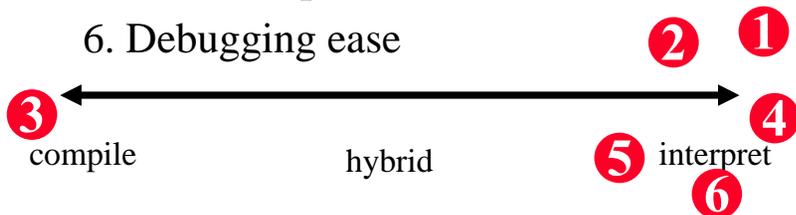
27

Programming Environments

- The collection of tools used in software development, often including an integrated editor, debugger, compiler, collaboration tool, etc.
- Modern Integrated Development Environments (IDEs) tend to be language specific, allowing them to offer support at the level at which the programmer thinks.
- Examples:
  - UNIX -- Operating system with tool collection
  - EMACS – a highly programmable text editor
  - Smalltalk -- A language processor/environment
  - Microsoft Visual C++ -- A large, complex visual environment
  - Your favorite Java environment: BlueJ, Jbuilder, J++, …
  - Generic: IBM's Eclipse

28

BlueJ: a simple Java IDE

http://www.bluej.org/

29
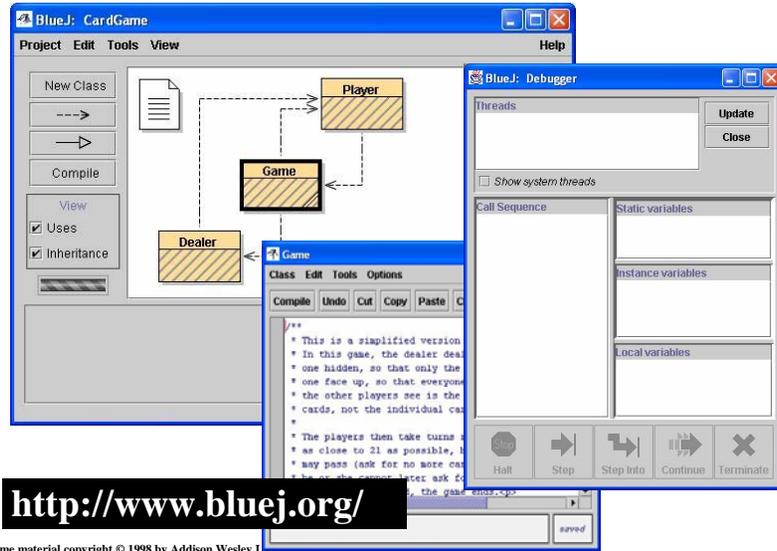
Summary

- Programming languages are constantly evolving
- There are many reasons to study the concepts underlying programming languages

30