Principles of Programming Languages

COMP251: Introduction

Prof. Dekai Wu

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong, China

Fall 2007
Hundreds of different PLs have been designed and implemented.

They may be grouped into different families of PLs.

We are not surveying PLs, but studying the programming concepts and constructs behind the different designs.

**Goal:**
- Improve your understanding of the language you are using.
- Systematically learn the various programming concepts and constructs.
- Help you learn a new language.
- Make it easier to design a new language.
- Allow a better choice of programming language.
How About Human Languages?

- **Chinese vs. English:**
  - pictorial (WYSIWYG) vs. phonetic
  - hieroglyphic vs. alphabetical

- **Japanese vs. English:**
  - wa-ta-shi-wa ni-hon-go wa-ka-ri ma-sen.
  - I Japanese understand don’t.

An intriguing question: Do the differences in human language designs reflect how differently people think?
Development of Human Languages

1st written language: Sumerian, 3500 B.C.
(c.f. Chinese, Shang Dynasty, 2000 B.C.)
1st alphabet: Phoenician, 1100 B.C.; only consonants.

<table>
<thead>
<tr>
<th>𐤁</th>
<th>𐤂</th>
<th>𐤃</th>
<th>𐤄</th>
<th>𐤅</th>
<th>𐤆</th>
<th>𐤇</th>
<th>𐤈</th>
</tr>
</thead>
<tbody>
<tr>
<td>hēt</td>
<td>zayin</td>
<td>wāw</td>
<td>hē</td>
<td>dālet</td>
<td>gīmel</td>
<td>bēt</td>
<td>'ālef</td>
</tr>
<tr>
<td>h</td>
<td>z</td>
<td>w</td>
<td>h</td>
<td>d</td>
<td>g</td>
<td>b</td>
<td>'</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>𐤉</th>
<th>𐤊</th>
<th>𐤋</th>
<th>𐤌</th>
<th>𐤍</th>
<th>𐤎</th>
<th>𐤏</th>
<th>𐤐</th>
</tr>
</thead>
<tbody>
<tr>
<td>sāmek</td>
<td>nun</td>
<td>mēm</td>
<td>lāmed</td>
<td>kāf</td>
<td>yōd</td>
<td>tēt</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>n</td>
<td>m</td>
<td>l</td>
<td>k</td>
<td>y</td>
<td>t</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>𐤑</th>
<th>𐤒</th>
<th>𐤓</th>
<th>𐤔</th>
<th>𐤕</th>
<th>𐤖</th>
<th>𐤗</th>
<th>𐤘</th>
</tr>
</thead>
<tbody>
<tr>
<td>tāw</td>
<td>šin/šin</td>
<td>rēš</td>
<td>qōf</td>
<td>šādē</td>
<td>pē</td>
<td>‘ayin</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>š</td>
<td>r</td>
<td>q</td>
<td>š</td>
<td>p</td>
<td>'</td>
<td></td>
</tr>
</tbody>
</table>
1st complete alphabet: Greek, 800 B.C.; consonants + vowels.

<table>
<thead>
<tr>
<th></th>
<th>Ionia</th>
<th>Athens</th>
<th>Corinth</th>
<th>Argos</th>
<th>Euboea</th>
<th>Modern</th>
<th>AP</th>
<th>MP</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td>ΑΑ</td>
<td>ΑΑ</td>
<td>ΑΑ</td>
<td>ΑΑ</td>
<td>ΑΑ</td>
<td>Αα</td>
<td>[a]</td>
<td>[a]</td>
</tr>
<tr>
<td>beta</td>
<td>ΒΒ</td>
<td>ΒΒ</td>
<td>ΒΒ</td>
<td>ΒΒ</td>
<td>ΒΒ</td>
<td>Ββ</td>
<td>[b]</td>
<td>[b]</td>
</tr>
<tr>
<td>gamma</td>
<td>ΓΓ</td>
<td>ΝΝ</td>
<td>ΓΓ</td>
<td>ΓΓ</td>
<td>ΓΓ</td>
<td>Γγ</td>
<td>[γ]</td>
<td>[γ]</td>
</tr>
<tr>
<td>delta</td>
<td>ΔΔ</td>
<td>ΔΔ</td>
<td>ΔΔ</td>
<td>ΔΔ</td>
<td>ΔΔ</td>
<td>Δδ</td>
<td>[δ]</td>
<td>[δ]</td>
</tr>
<tr>
<td>epsilon</td>
<td>ΕΕ</td>
<td>ΕΕ</td>
<td>ΕΕ</td>
<td>ΕΕ</td>
<td>ΕΕ</td>
<td>Εε</td>
<td>[ε]</td>
<td>[ε]</td>
</tr>
<tr>
<td>digamma</td>
<td>ΤΤ</td>
<td>ΤΤ</td>
<td>ΤΤ</td>
<td>ΤΤ</td>
<td>ΤΤ</td>
<td>[τ]</td>
<td>[τ]</td>
<td>[τ]</td>
</tr>
<tr>
<td>zeta</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>Ηζ</td>
<td>[ζ]</td>
<td>[ζ]</td>
</tr>
<tr>
<td>eta</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>Ηη</td>
<td>[η]</td>
<td>[η]</td>
</tr>
<tr>
<td>theta</td>
<td>ΘΘ</td>
<td>ΘΘ</td>
<td>ΘΘ</td>
<td>ΘΘ</td>
<td>ΘΘ</td>
<td>Θθ</td>
<td>[θ]</td>
<td>[θ]</td>
</tr>
<tr>
<td>iota</td>
<td>ΙΙ</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>ΗΗ</td>
<td>Ωι</td>
<td>[ι]</td>
<td>[ι]</td>
</tr>
<tr>
<td>kappa</td>
<td>ΚΚ</td>
<td>ΚΚ</td>
<td>ΚΚ</td>
<td>ΚΚ</td>
<td>ΚΚ</td>
<td>Κκ</td>
<td>[κ]</td>
<td>[κ]</td>
</tr>
<tr>
<td>lambda</td>
<td>ΛΛ</td>
<td>ΛΛ</td>
<td>ΛΛ</td>
<td>ΛΛ</td>
<td>ΛΛ</td>
<td>Λλ</td>
<td>[λ]</td>
<td>[λ]</td>
</tr>
<tr>
<td>mu</td>
<td>ΜΜ</td>
<td>ΜΜ</td>
<td>ΜΜ</td>
<td>ΜΜ</td>
<td>ΜΜ</td>
<td>Μµ</td>
<td>[μ]</td>
<td>[μ]</td>
</tr>
<tr>
<td>nu</td>
<td>ΝΝ</td>
<td>ΝΝ</td>
<td>ΝΝ</td>
<td>ΝΝ</td>
<td>ΝΝ</td>
<td>Nν</td>
<td>[ν]</td>
<td>[ν]</td>
</tr>
<tr>
<td>xi</td>
<td>ΞΞ</td>
<td>ΞΞ</td>
<td>ΞΞ</td>
<td>ΞΞ</td>
<td>ΞΞ</td>
<td>Ξξ</td>
<td>[ξ]</td>
<td>[ξ]</td>
</tr>
<tr>
<td>omicron</td>
<td>ΩΩ</td>
<td>ΩΩ</td>
<td>ΩΩ</td>
<td>ΩΩ</td>
<td>ΩΩ</td>
<td>Ωο</td>
<td>[ο]</td>
<td>[ο]</td>
</tr>
<tr>
<td>pi</td>
<td>ΠΠ</td>
<td>ΠΠ</td>
<td>ΠΠ</td>
<td>ΠΠ</td>
<td>ΠΠ</td>
<td>Ππ</td>
<td>[π]</td>
<td>[π]</td>
</tr>
<tr>
<td>san</td>
<td>ΜΜ</td>
<td>ΜΜ</td>
<td>ΜΜ</td>
<td>ΜΜ</td>
<td>ΜΜ</td>
<td>[σ]</td>
<td>[σ]</td>
<td>[σ]</td>
</tr>
<tr>
<td>kappa</td>
<td>ΞΞ</td>
<td>ΞΞ</td>
<td>ΞΞ</td>
<td>ΞΞ</td>
<td>ΞΞ</td>
<td>[σ]</td>
<td>[σ]</td>
<td>[σ]</td>
</tr>
<tr>
<td>rho</td>
<td>ΡΡ</td>
<td>ΡΡ</td>
<td>ΡΡ</td>
<td>ΡΡ</td>
<td>ΡΡ</td>
<td>Ρρ</td>
<td>[ρ]</td>
<td>[ρ]</td>
</tr>
<tr>
<td>sigma</td>
<td>ΣΣ</td>
<td>ΣΣ</td>
<td>ΣΣ</td>
<td>ΣΣ</td>
<td>ΣΣ</td>
<td>Σσ</td>
<td>[σ]</td>
<td>[σ]</td>
</tr>
<tr>
<td>tau</td>
<td>ΤΤ</td>
<td>ΤΤ</td>
<td>ΤΤ</td>
<td>ΤΤ</td>
<td>ΤΤ</td>
<td>Ττ</td>
<td>[τ]</td>
<td>[τ]</td>
</tr>
<tr>
<td>upsilon</td>
<td>ΨΨ</td>
<td>ΨΨ</td>
<td>ΨΨ</td>
<td>ΨΨ</td>
<td>ΨΨ</td>
<td>Ψλ</td>
<td>[λ]</td>
<td>[λ]</td>
</tr>
<tr>
<td>phi</td>
<td>ΦΦ</td>
<td>ΦΦ</td>
<td>ΦΦ</td>
<td>ΦΦ</td>
<td>ΦΦ</td>
<td>Φφ</td>
<td>[φ]</td>
<td>[φ]</td>
</tr>
<tr>
<td>chi</td>
<td>ΧΧ</td>
<td>ΧΧ</td>
<td>ΧΧ</td>
<td>ΧΧ</td>
<td>ΧΧ</td>
<td>Χχ</td>
<td>[χ]</td>
<td>[χ]</td>
</tr>
<tr>
<td>psi</td>
<td>ΨΨ</td>
<td>ΨΨ</td>
<td>ΨΨ</td>
<td>ΨΨ</td>
<td>ΨΨ</td>
<td>Ψψ</td>
<td>[ψ]</td>
<td>[ψ]</td>
</tr>
<tr>
<td>omega</td>
<td>ΩΩ</td>
<td>ΩΩ</td>
<td>ΩΩ</td>
<td>ΩΩ</td>
<td>ΩΩ</td>
<td>Ωω</td>
<td>[ω]</td>
<td>[ω]</td>
</tr>
</tbody>
</table>
What’s a PL for?

Stroustrup (C++ designer, 1994):

- tool for instructing machines?
- means for communicating between programmers?
- vehicle for expressing high level designs?
- notation for algorithms?
- way of expressing relationships between concepts?
- tool for experimentation?
- means for controlling computerized devices?
- collection of “neat” features?

His answer: All of the above except the last one.
main:
    !PROLOGUE# 0
    save %sp,-128,%sp

    !PROLOGUE# 1
    mov 1,%o0
    st %o0,[%fp-20]
    mov 2,%o0
    st %o0,[%fp-24]
    ld [%fp-20],%o0
    ld [%fp-24],%o1
    add %o0,%o1,%o0
    st %o0,[%fp-28]
    mov 0,%i0
    nop
#include <stdio.h>

int main()
{
    int x, y, z;

    x = 1;
    y = 2;
    z = x+y;

    return 0;
}
Levels of PLs

- **machine (binary) language** is unintelligible
- **assembly language** is low level
  - mnemonic names for machine operations
  - explicit manipulation of memory addresses/contents
  - machine dependent
- **high level language**
  - readable
    - instructions are easy to remember
    - faster coding
    - less error-prone (fewer bugs?)
    - easier to maintain
  - no mention of memory locations
  - machine independent = portable
Genealogy of Common PLs

- Fortran I
- Fortran II
- ALGOL 58
- FLOW-MATIC
- COBOL
- APL
- SIMULA I
- BASIC
- LISP
- CPL
- SNOBOL

1957
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
00
01
02

FORTRAN 77

MODULA-2

Pascal

Modula-3

AWK

Smalltalk 80

Ada 83

Perl

QuickBASIC

Eiffel

Visual BASIC

Java

JavaScript

Python

C#

C++

Common Lisp

Haskell

Miranda

ICON

ML

Scheme

C

PROLOG

Fortran IV

Fortran

Fortran 90

Fortran 95
Genealogy of Common PLs (Sethi 1996)
4 Paradigms of PL Design

- Imperative Programming (IP) or Procedural Programming (PP)
  - See http://en.wikipedia.org/wiki/Procedural_programming
- Object-Oriented Programming (OOP)
- Declarative Programming
  - Functional Programming (FP)
  - Logic Programming (LP)

PL design is a balance among:
- efficiency
- readability
- support
- taste!
### IP/PP, OOP, FP, LP

<table>
<thead>
<tr>
<th>IP/PP</th>
<th>OOP</th>
<th>FP</th>
<th>LP</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN</td>
<td>Smalltalk</td>
<td>LISP</td>
<td>Prolog</td>
</tr>
<tr>
<td>Pascal</td>
<td></td>
<td>Scheme</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Java</td>
<td>SML</td>
<td></td>
</tr>
<tr>
<td>action-oriented</td>
<td>object-oriented</td>
<td>function-oriented</td>
<td>logic-oriented</td>
</tr>
<tr>
<td>procedural assignments</td>
<td>classes</td>
<td>functions</td>
<td>logic reasoning</td>
</tr>
<tr>
<td>algorithm design</td>
<td>system building</td>
<td>mapping: $x \rightarrow f(x)$</td>
<td>“Are you sick?”</td>
</tr>
<tr>
<td>compile</td>
<td>compile</td>
<td>interpret</td>
<td>interpret</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- FORTRAN, Pascal, C
- Smalltalk, Java, C++, Scheme, SML
- action-oriented, object-oriented, function-oriented, logic-oriented
- procedural assignments, classes, inheritance
- system building, reusable software
- algorithm design
- compile, compile
- interpret
- “Are you sick?”
- expert system, database queries
A **compiler** translates source programs into machine codes that run directly on the target computer. e.g. a.c → a.out.

- static codes
- compile once, run many
- optimized codes
  ⇒ more efficient
- examples: FORTRAN, Pascal, C++
An **interpreter** is a virtual machine implemented on a target computer which runs a source program directly.

- slower
- interpret many, run many
- interactive mode: easy debugging
- more flexible: allow programs to be changed “on the fly”
- examples: many script languages (sh, csh, tcl, awk), ML, PROLOG
A hybrid system translates high-level source programs to an intermediate language which then allows fast and easy interpretation.

- compile once,
  interpret many

Examples: UCSD Pascal, Perl, Python, Java
There are hundreds of different PLs.
It is easier to write (large) programs with a high-level PL.
Will emphasize the basic programming concepts/constructs.
Will address 4 programming paradigms: IP/PP, OOP, FP, LP.
2 approaches to types within the FP paradigm: latent typing vs. static typing with type inference.
2 ways to implement PLs: compilation vs. interpretation.