CS 603: Programming Language Organization

Lecture 1 Spring 2005 Department of Computer Science University of Alabama Joel Jones

Outline

- Prerequisites
- Personal Information
- Motivation
- ImpCore

Prerequisites

- Knowledge of several different programming languages
- At least one non-imperative

Teaming and Active Learning

- Active Participation in Classroom
 - Questioning
 - Preparation
 - Working in Pairs
 - Reporting

Why Study Weird Programming Languages?

• Why don't you tell me?

Pair Up:

- Why study weird programming languages?
- Why are you taking this class?

Languages and their Paradigms

- Brainstorm—name as many programming languages as you can.
 - LISP, Scheme, CLOS, Python, APL, C, C++,
 Smalltalk, Ruby, Perl, AWK, COBOL, FORTRAN,
 AWK, Pascal, AppleScript, Visual Basic, BASIC, Java,
 Prolog, CLP, SML, Haskell, etc.
- Brainstorm—what "paradigms" or "kinds" of languages are these?
 - functional (applicative), object-oriented, imperative, scripting, logic programming

Personal Information		
Name:	Email:	
Programming Languages/Environments:		
Group Work Experience:		
What you expect of the class:		
	Laptop?	
	©2005 Joel Jones	



What is the Simplest Possible Language?

- Simple Syntax
- Few Primitives

ImpCore Language

- Two kinds of expressions
 - Function definitions:
 - (define double (x) (+ x x))
 - Expressions:
 - (double 5)
- Environment is interactive/interpreted

ImpCore Language

- Four kinds of elements at top-level
 - Expressions:
 - (double 5)
 - Function definitions:
 - (define double (x) (+ x x))
 - Variable definitions
 - (val x 5)
 - Source Import
 - (use double.imp)
- Environment is interactive/interpreted

LISP-like Syntax

toplevel ::= exp (use file-name) (val variable-name exp)	<pre>integer ::= sequence of digits, possibly prefixed with a plus or minus sign</pre>
<pre>exp (val val lable name exp) i (define function-name (formals) exp) exp ::= value</pre>	<pre>*-name ::= sequence of characters not an integer and not containing (,), ;, or whitespace</pre>
<pre>formals ::= {variable-name} value ::= integer function ::= function-name</pre>	
primitive primitive ::= + _ * / = < > print	