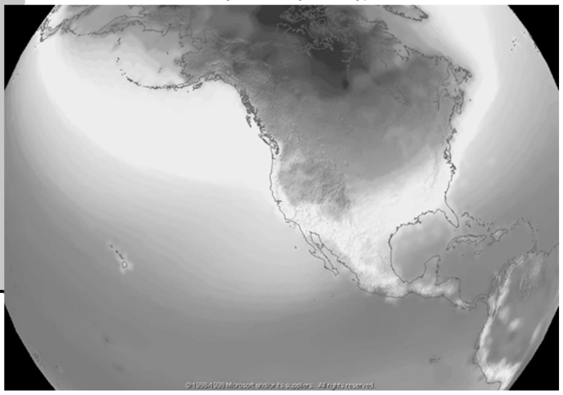
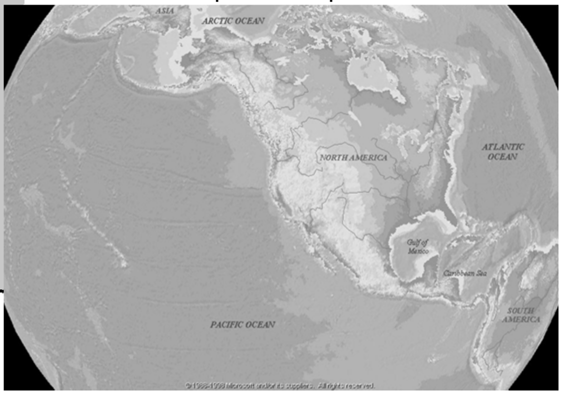
CSIS 4244 Programming Language Concepts

An Introduction

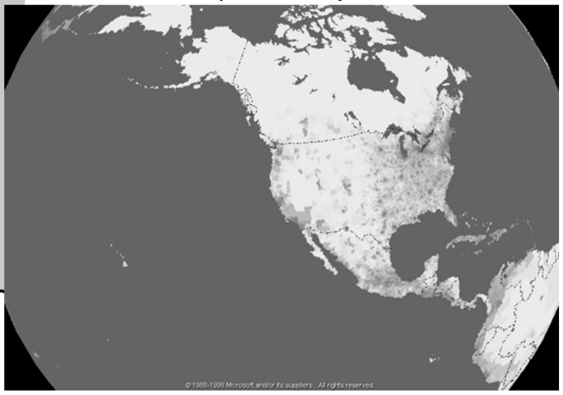
Temperature (January)


© 1988-1998 Microsoft and/or its suppliers. All rights reserved.

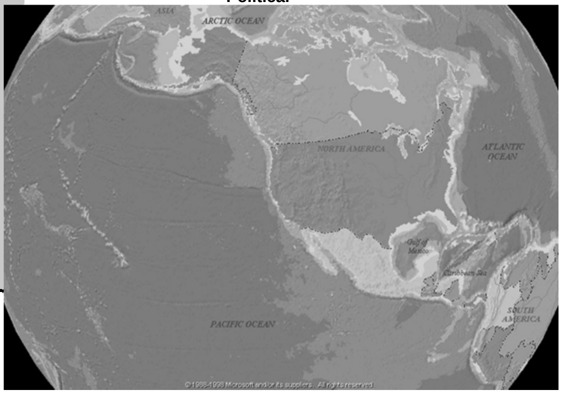
Comprehensive Map



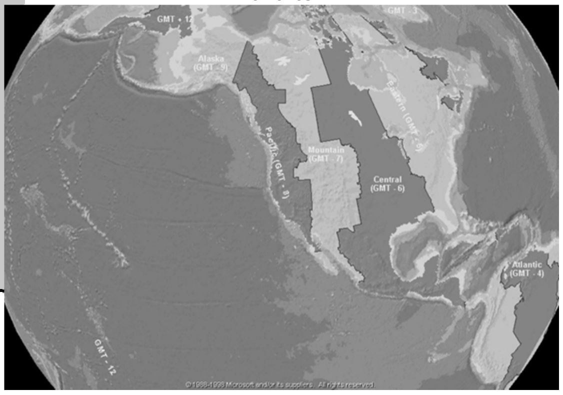
Population Density


© 1988-1998 Microsoft and/or its suppliers. All rights reserved.

Political



Time Zones



Road Map

What is a programming language?

- A communication interface between human and machine
- An abstraction of the real world (user’s view)
- A notation for expressing objects & algorithms (programmer’s view)
- The set of all syntactically correct programs (compiler’s view)

So What’s the Point?

- The earth has a lot of “standard” features (rivers, land masses, etc.) which can be *simulated* on maps
- Different kinds of maps provide different kinds of information, but…
- It is necessary to understand the underlying concepts of maps to effectively use them
  - What kind of symbols are allowed & what are the rules for their use? (*syntax*)
  - What do the symbols mean? (*semantics*)
  - What kind of map is best suited for a particular purpose? (*pragmatics*)

Elements of Programming Languages

- Syntax
  - Formal rules for producing grammatically correct program constructs
- Semantics
  - Rules governing the meaning (operational effect) of syntactically correct program constructs when translated and executed
- Pragmatics
  - What kind of objects (data) a language can manipulate
  - What kind of algorithms it can (reasonably) implement

CSIS 4244

- This course covers the underlying concepts of contemporary programming languages
- This is NOT a survey of languages course
  - But we will see many examples of how various languages use the underlying concepts
- This is NOT a programming course
  - But there will be short programming assignments in various languages

Reasons to Study Programming Languages Concepts

1. Better able to choose appropriate languages
2. Increased ability to learn new languages
3. Increased capacity to use language concepts
   a. Understand obscure features
   b. Understand implementation costs
   c. Work around limitations
4. Increased ability to design new languages

Programming Domains

1. Scientific applications
2. Business applications
3. Artificial intelligence
4. Systems programming
5. Scripting languages
6. Special purpose languages

What Makes a Language Successful?

- Easy to learn (BASIC, Pascal, Scheme)
- Easy to express things, easy to use once fluent, "powerful" (C, Lisp, APL, Algol-68, Perl)
- Easy to implement (BASIC)
- Can compile to fast/small code (Fortran)
- Backing of a powerful sponsor (COBOL, PL/1, Ada, Visual Basic)
- Wide dissemination at minimal cost (Pascal, Java)

Language Evaluation Criteria

1. *Readability*
   - *Factors:*
     - Overall simplicity
       Too many features and multiplicity of features are bad
     - Control statements
       Data type and structures
       Syntax considerations

2. *Writeability*
   - *Factors:*
     - Simplicity
     - Support for abstraction
     - Expressivity

APL Readability?

- The following APL program produces the prime numbers in the range from 1 to N:

```
(2=(+/[2]0=(ιN)°.|(ιN)))/ιN
```

Evaluation criteria

3. *Reliability*
   - *Factors:*
     - Type checking
     - Exception handling
     - Aliasing
     - Readability and writeability
4. *Cost*
   - *Factors*
     - Programmer training, software creation
     - Compilation, Execution
     - Poor reliability
     - Maintenance

Language Categories

Imperative
- Computation viewed as actions that manipulate memory locations
- Fortran/C/Basic

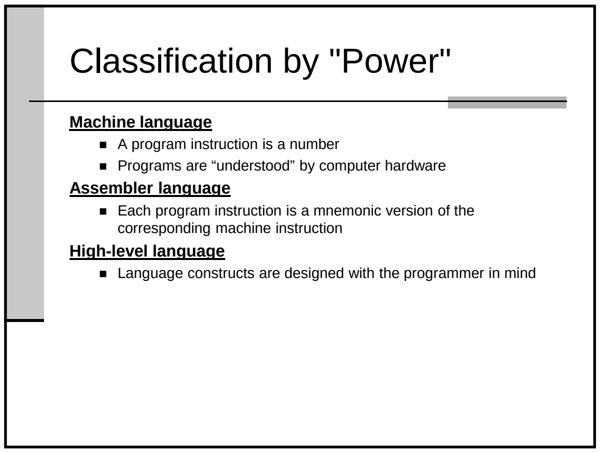
Functional
- Emphasizes function evaluations
- Lisp/ML/Scheme/F#

Object Oriented
- Hierarchical organization of data, operations applied to objects
- Java/C#/Smalltalk/Ruby

Logic
- Uses “logic” as a means of specifying computation (what vs. how)
- Prolog

Scripting
- "Glue" languages, extension languages, batch processing
- Tcl/Python/Ruby

Concurrent
- Code is executed in parallel. Multiple operations occur “simultaneously”
- Java/Ada/Erlang

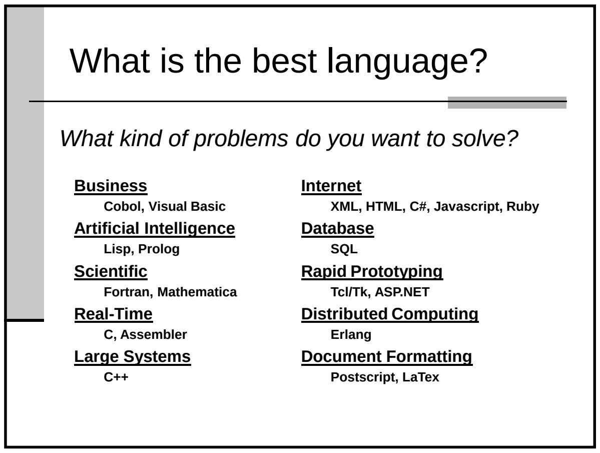


Classification by "Power"

Machine language
- A program instruction is a number
- Programs are "understood" by computer hardware

Assembler language
- Each program instruction is a mnemonic version of the corresponding machine instruction

High-level language
- Language constructs are designed with the programmer in mind



What is the best language?

What kind of problems do you want to solve?

Business
- Cobol, Visual Basic

Artificial Intelligence
- Lisp, Prolog

Scientific
- Fortran, Mathematica

Real-Time
- C, Assembler

Large Systems
- C++

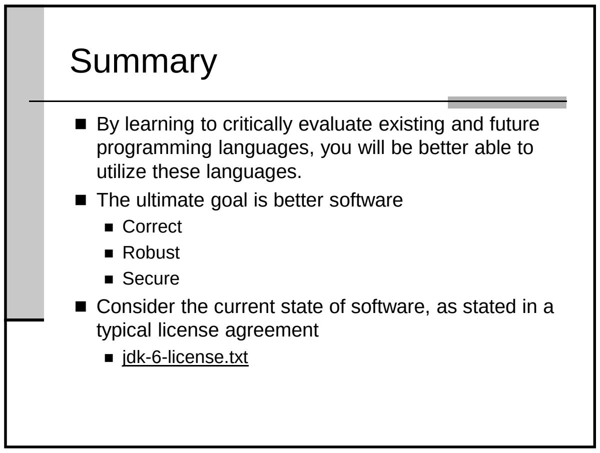
Internet
- XML, HTML, C#, Javascript, Ruby

Database
- SQL

Rapid Prototyping
- Tcl/Tk, ASP.NET

Distributed Computing
- Erlang

Document Formatting
- Postscript, LaTex



Summary

- By learning to critically evaluate existing and future programming languages, you will be better able to utilize these languages.
- The ultimate goal is better software
  - Correct
  - Robust
  - Secure
- Consider the current state of software, as stated in a typical license agreement
  - jdk-6-license.txt