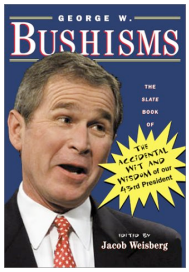
Language Issues



Misunderstimated?
Sublimable?
Hopefuller?

"I know how hard it is for you to put food on your family."

"I know the human being and fish can coexist peacefully."

Outline

- Announcements:
- Selecting a Language
- FORTRAN
- C
- MATLAB
- Java

When we last saw our heros…

- By now, you've
  - designed your program
  - specified important sections
- You are now ready to code
  - but what language will you choose?

Language Criteria

- You may not have a choice
  - If you are extending a program written in C, use C!
- You may only know one language
  - This is not a reason, it's an excuse!
  - But, there's a lot to be gained by sticking with what you know

Language Criteria

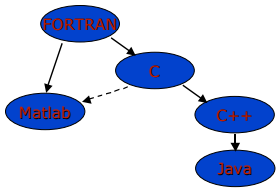
- Several things to keep in mind when picking a language
  - Libraries and legacy code: Can you easily utilize code that is already written, tested & debugged (& hopefully specified)?
  - Portability: is the language standardized so that it easy to compile and run on several platforms?
  - Elegance: will your program be easier to understand (and debug and extend) in one language
  - Future: will compilers still be available in 10 years? Will future users be comfortable in this language?

Languages for Scientific Computing

- Programming languages can be categorized
  - Procedural: Programs consist of one or more procedures (aka functions or subroutines). Data objects are passed to the procedures (FORTRAN, C, Matlab)
  - Object-Oriented: Programs are composed of several objects that encapsulate related data and the procedures to manipulate them (C++, Java)
  - Functional: Programs are functions that operate on data or other functions--highly recursive (LISP, ML)

Languages for Scientific Computing

- Language phylogeny



FORTRAN

- FORmula TRANslator
  - One of the first programming languages
  - Most common strain was standardized in 1977
  - Designed for "Scientific Computing" (e.g. physics)

FORTRAN

- Types:
  - integer, float, double, complex, char
- Data structures:
  - arrays

FORTRAN: Key Advantages

- complex type fully implemented, integrated
- lots of legacy code
- simple
- fast!

FORTRAN: Disadvantages

- F77 is ancient
  - Missing "modern" features like pointers, novel data structures (or even records)
  - Missing not-so-modern features like recursion!
  - Encourages bad programming:
    - heavy use of goto-statement
    - common blocks

FORTRAN90

- Modernizes F77 while maintaining backward compatability
  - Dynamic allocation of arrays (size set at run-time, not at compilation)
  - "Vectorized" operations:
    - c= a(:)+b(:)
- I'm not a fan (just added stuff to F77), but some folks really like it
  - An attractive option for extending legacy code

C

- In many ways, C is similar to FORTRAN
  - Procedural
  - few built-in types and data structures
- But more modern
  - pointers--allows you to manipulate memory directly
  - structs--allows you to implement records
  - Together, pointers and structs allow you to create new data structures
  - supports recursion

C: Key advantages

- Common--good compilers available for all platforms, often for free
- Legacy code--lots of stuff already written
- Good performance--comparable for FORTRAN, especially for simple, array-based code
- Very modular--library-concept tightly integrated through #include directive
- Modern--can do everything you could ever want to do (math, CS, graphics)

C: Key disadvantages

- Programming with pointers can be complicated and even dangerous
- No complex type or functions

Matlab

- Matlab is a "programming environment" for scientific computing
  - Lots of built-in functions
  - Easy to program, especially if you are comfortable with procedural programming
  - Data analysis/visualization tools make it easy to develop/debug code
  - Excellent system for building prototypes, but not suitable for production runs of large, computationally intensive code

Java

- Java is the current standard for object-oriented programming
  - objects are "classes"
    - fields: data
    - methods: functions
  - classes should encompass data-types with functions that operate on them

Java: Key Advantages

- Object-oriented--encapsulation of data and functions simplifies programs, makes management easier
- Popular--lots of available code, especially for graphics and common CS algorithms and data structures
- Standardized--very complete specification of language means that all Java code will run on all Java compilers
  - Several versions though, make sure your compilers is current
  - Be wary of Microsoft who is trying to create a proprietary version!
- Strongly-typed--many bugs are caught at compile time
- Run-time checks on array bounds avoids "segmentation faults" (returns an intelligent error message)

Java: Key Disadvantages

- Performance: object-oriented languages are complicated, so it is hard for compilers to make smart optimizations (applies to C++, too)
  - Also, standard java is interpreted, not compiled so optimization is out of the question!
- Main Audience: scientists are no longer the main driving force behind computers.
  - Java's main audience are commercial developers (especially web)
  - Even so, there is still a lot of scientific code out there

My Advice

- For extending old code, stick with the original language
- For new code, I highly recommend Java
  - code is cleaner, and requires less debugging (reduces development time)
  - Especially true if your work doesn't require heavy computation
- For any project, consider developing a prototype first (Matlab or Java) and then translate to C or FORTRAN for max performance