Outline

- Announcements
  - HWI due on Friday
- Differences between FORTRAN and C
- Calling FORTRAN from C

Comparing C and FORTRAN

FORTRAN

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

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
  - 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
- LESS LEGACY CODE!
  - Calling this old code from C would allow us to have the best of both worlds!

Calling FORTRAN from C

- In theory, we should be able to
  - Compile FORTRAN code to object code (-c option)
  - Compile C code to object code
  - Link objects together
- However, there are a few wrinkles:
  - Namespace problem
    - C needs to refer to the routines using the correct names
    - ANSI C code needs prototypes
  - Call-by-value problem
    - C can use call-by-value, FORTRAN uses only call-by-reference
    - In general, need to make sure we're sending the FORTRAN routines the type of data they expect

Namespace Problem

- The section of a .o file for a specific routine is given a name.
- The name is used by the linker to figure out how the executable is put together
- We must ensure that calls to FORTRAN routines in C object code use the same name as in the FORTRAN .o file

Namespace Problem

- Routine FooBar in a FORTRAN .o file could be
  - FooBar_
  - FOOBAR_
  - foobar_ (g77)
- To call FooBar from C, you will need to use the correct case and add the underscore
  - Some compilers provide a -f option which forces the names in the .o to be all lower case
- CAUTION: Every system/compiler is different! Read the documentation!

Call-by-Value Problem

- In C, a variable can be passed to a subroutine by value or reference.
  - call-by-value: the number stored in the variable is passed to the subroutine. The value in the calling routine WILL NOT CHANGE!
    - int m = 4
    - Foo(m); /* m won't change */
    - prototype for Foo:
      - void Foo(int m);

Call-by-reference:

- call-by-reference: the memory address is passed. If the subroutine modifies the value, the value WILL CHANGE in the calling routine.
  - Use "&" to pass a scalar by value:
    - Foo(&m) /* m might change */
    - prototype for Foo:
      - » void Foo(int *m); /* "*"==pointer */
  - Arrays are already pointers, so they are automatically passed by reference:
    - int m[10],tot;
    - tot=SumArray(m,10);
    - prototype for SumArray:
      - » int Foo(int *m, int n); /* n=length(m) */

Type Equivalences

FORTRAN	C
character*n c	char c[n]
integer (integer *4)	int
real (real *4)	float
double (real *8)	double
complex *16 c	struct dcomp{ double real; double cmplx; };struct dcomp c;

Multidimensional Arrays

- 1D arrays of equivalent type are represented identically by C and FORTRAN
  - not true for multidimensional arrays

A=

1	2	3
4	5	6

C: Column-major

1
2
3
4
5
6

FORTRAN: Row-major

1
4
2
5
3
6