EECE 276
Embedded Systems

Survey: Ada 95, C/C++, C#
Ada

- DoD’s response to the “software crisis”
  - Ada 83 and Ada 95
  - Originally: for embedded real-time systems
  - Mandated by DoD until 1996, still popular in EU

- Support for…
  - Large-scale development: packages
  - OO: “tagged types”
  - Real-time systems:
    - Tasks
    - Protected units
    - Rendezvous (synchronization)
    - Distributed extensions
C

- Low-level language
  - “Machine-independent assembly”
- Special markers affecting code generation
  - register, static: allocation
  - volatile: not optimized away by compiler
- Automatic type coercion (float -> int)
- Exception “handling”
  - setjmp/longjmp
- Efficient code, very little safety.
C++

- Hybrid: C with OO capabilities
- Preprocessing (text macros) – weak type checking
- “Compilation units” : Header and source files
- Main source of problems:
  - Memory approach (pointers, etc.)
- OO-style: classes, structs, unions
  - Member variables and member functions
  - Access control (public/private/protected)
- Templates:
  - “Typed” classes and functions
- For real-time systems?
  - Overhead?
  - Concurrency issues?
  - Memory management?
C#

- C++- and Java-like
- Uses a VM for execution (.NET CLR)
- OO language with most concepts supported
  - classes, interfaces
  - Memory management is compatible with RT needs
- Threads and synchronization constructs
  - Lock, monitors, mutex, interlock (priority inheritance)
  - Timers
- May not be appropriate for hard RT systems (schedulability and determinism), but may work for soft and firm RT systems
Fortran

- Oldest higher-order language, still used in specific areas (e.g. physics, numerics, etc.)
- In (historical) RT systems:
  - Needed assembly code to manage HW
- New developments:
  - Parallel programming constructs
  - Very efficient optimizing compilers
- Still used in legacy systems and in coding portions of new systems
- Not a “real-time” language...