Type Checking Parameters

- Considered very important for reliability
- FORTRAN 77 and original C: none
- Pascal, FORTRAN 90, Java, and Ada: it is always required
- ANSI C and C++: choice is made by the user
  - Prototypes
- Relatively new languages Perl, JavaScript, and PHP do not require type checking
- In Python and Ruby, variables do not have types (objects do), so parameter type checking is not possible
Design Considerations for Parameter Passing

• Two important considerations
  – Efficiency
  – One-way or two-way data transfer
• But the above considerations are in conflict
  – Good programming suggest limited access to variables, which means one-way whenever possible
  – But pass-by-reference is more efficient to pass structures of significant size
Parameters that are Subprogram Names

- It is sometimes convenient to pass subprogram names as parameters

- Issues:
  1. Are parameter types checked?
  2. What is the correct referencing environment for a subprogram that was sent as a parameter?
Function Parameter Type Checking

• C and C++: functions cannot be passed as parameters but pointers to functions can be passed and their types include the types of the parameters, so parameters can be type checked
• FORTRAN 95 type checks
• Ada does not allow subprogram parameters; an alternative is provided via Ada’s generic facility
• Java does not allow method names to be passed as parameters
Referencing Environment

- **Shallow binding**: The environment of the call statement that enacts the passed subprogram
  - Most natural for dynamic-scoped languages
- **Deep binding**: The environment of the definition of the passed subprogram
  - Most natural for static-scoped languages
- **Ad hoc binding**: The environment of the call statement that passed the subprogram