Pass-by-Value (In Mode)

- The value of the actual parameter is used to initialize the corresponding formal parameter
  - Normally implemented by copying
  - Can be implemented by transmitting an access path but not recommended (enforcing write protection is not easy)
  - **Disadvantages** (if by physical move): additional storage is required (stored twice) and the actual move can be costly (for large parameters)
  - **Disadvantages** (if by access path method): must write-protect in the called subprogram and accesses cost more (indirect addressing)

Pass-by-Result (Out Mode)

- When a parameter is passed by result, no value is transmitted to the subprogram; the corresponding formal parameter acts as a local variable; its value is transmitted to caller’s actual parameter when control is returned to the caller, by physical move
  - Require extra storage location and copy operation
- **Potential problem**: `sub(p1, p1);` whichever formal parameter is copied last will represent the current value of `p1`

Pass-by-Value-Result (Inout Mode)

- A combination of pass-by-value and pass-by-result
- Sometimes called **pass-by-copy**
- Formal parameters have local storage
- **Disadvantages**:
  - Those of pass-by-result
  - Those of pass-by-value
Pass-by-Reference (Inout Mode)

- Pass an access path
- Also called pass-by-sharing
- **Advantage:** Passing process is efficient (no copying and no duplicated storage)
- **Disadvantages:**
  - Slower accesses (compared to pass-by-value) to formal parameters
  - Potentials for unwanted side effects (collisions)
  - Unwanted aliases (access broadened)

Pass-by-Name (Inout Mode)

- By **textual** substitution
- Formals are bound to an access method at the time of the call, but actual binding to a value or address takes place at the time of a reference or assignment
- Allows flexibility in late binding
- Will see examples in the next lecture

Implementing Parameter-Passing Methods

- In most languages, parameter communication takes place thru the run-time stack
- Pass-by-reference are the simplest to implement; only an address is placed in the stack
- A subtle but fatal error can occur with pass-by-reference and pass-by-value-result: a formal parameter corresponding to a constant can mistakenly be changed

Parameter Passing Methods of Major Languages

- **C**
  - Pass-by-value
  - Pass-by-reference is achieved by using pointers as parameters
- **C++**
  - A special pointer type called reference type for pass-by-reference
- **Java**
  - All parameters are passed by value
  - Object parameters are passed by reference
- **Ada**
  - Three semantics modes of parameter transmission: in, out, in out; in is the default mode
  - Formal parameters declared out can be assigned but not referenced; those declared in can be referenced but not assigned; in out parameters can be referenced and assigned
Parameter Passing Methods of Major Languages (cont’d)

- **Fortran 95**
  - Parameters can be declared to be in, out, or inout mode
- **C#**
  - Default method: pass-by-value
  - Pass-by-reference is specified by preceding both a formal parameter and its actual parameter with `ref`
- **PHP**: very similar to C#
- **Perl**: all actual parameters are implicitly placed in a predefined array named `@_
- **Python** and **Ruby** use pass-by-assignment (all data values are objects)

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

Multidimensional Arrays as Parameters

- If a multidimensional array is passed to a subprogram and the subprogram is separately compiled, the compiler needs to know the declared size of that array to build the storage mapping function

Multidimensional Arrays as Parameters: C & C++

- Programmer is required to include the declared sizes of all but the first subscript in the actual parameter
- Disallows writing flexible subprograms
- Solution: pass a pointer to the array and the sizes of the dimensions as other parameters; the user must include the storage mapping function in terms of the size parameters