Matching Happens on a Per-Argument Basis

How does a compiler choose?

The **basic rule**:
- pick the **"most derived" class**
- **for each argument**.

The result is **not always unique**, as you already know.

Compilers report **errors if** the **results** are **ambiguous**.

Pitfall: Overloading too Finely Can Confuse Users

Exact rules exist, but don't seem to be widely known, understood, portable, and so forth.*

My advice: **as with precedence,**
- **if you don't know the answer,**
- **don't try to look it up**:
- instead, avoid using it.

Matching overloaded calls is much more dangerous—one can 'steal' calls from existing code by creating new functions that provide better matches.

*See ECE409 notes L7P5-6 for more detailed comments, or the C++ ARM or standards for a definitive version.

Some Operators Cannot be Overloaded

C++ allows most, but not all, operators to be overloaded.

Operators that cannot be overloaded include
- member access ("`.`"),
- pointer to member function invocation ("`.*`"),
- conditional expressions ("`? :`"), and
- scope identification ("`::`").

C Equivalences May Not Be Valid in C++

In C++, **C's equivalences may no longer hold**.

For example, **pointer-like and array-like objects are not necessarily the same**
- `array[10]` (calls `operator[]`)
- may not be the same as
- `*(array + 10)` (calls `operator+` and `operator*`).

These operators **can be defined to be consistent**, however.

