Function resolution in C++
The problem -- which of several inexact matches is closest to the called function

- The crux of the problem is how to define “closest”
- Java and C++ use different definitions
Java

• A three step process
  • find all functions that could match the call
  • eliminate functions whose parameter types can be obtained from widening another functions parameter types
  • if one function left, call it, otherwise give an error
class account {
  private:
    double balance;
  public:
    account (double b) { balance = b; }
    operator double (void) { return balance; }
};

int main (void) {
  account acc(100.0);
  double balance = acc; // account to double conversion
  std::cout << balance << std::endl;
  return 0;
}
Object to Object conversion

• Assume two classes X and Y.

• A constructor X(Y yArg) {...} can be used to convert a Y object to an X object

• Consider the code:

  Y y = Y(1.0)

  X x = y

• The second line uses the X(Y) constructor to do the conversion from y to x.

• This will happen automatically if you don’t use the explicit keyword
Object to Object conversion

`explicit X(Y yArg) { ... }`

`Y y = Y(1.0)`

`X x = (X) y`

• The explicit keyword requires the program to request an explicit conversion

• `foo((X) y)` can call a `foo(X)` function, but cast is needed because of explicit

• In general, always use explicit

• `cout << i << endl;` is an example of implicit conversion that we use a lot.
implicit and explicit conversions

• Having user defined conversions happen randomly can be dangerous
  • typos assigning classes to primitives become silent conversions
  • function call matching gets harder to figure out
  • use explicit
C++

• Five levels of specificity of matching:

1. the two types agree after *trivial type conversions*, e.g., pointer to array type, T to const T

2. type conversion requires only a *promotion*, e.g., integral types stay integral, non-integral types stay non-integral, there is no loss of information

3. type conversion requires a *standard conversion*. Examples of this are int to double, double to int, T* to void*, Derived to Base

4. Requires a user defined type conversion

5. Match using ellipsis
Overload resolution finds the function that requires the lowest level of specificity to match. If two or more functions have the same lowest level, and the same number of matches at that level, it is an error. Otherwise, the function that matches the most arguments at the lowest level is returned.