Forwarding References Solutions

Nested References

- Is it possible to create a reference to a reference?
  - Programmers cannot write code that does this
  - The compiler can do it internally, for a type alias or a template parameter
- Write a simple program to illustrate your answer

Reference Collapsing Rules

- What is meant by "reference collapsing"?
  - When the compiler encounters a reference to a reference, it "collapses" the multiple references into a simple reference
- What rules are used for reference collapsing in C++11?
  - An rvalue reference to an rvalue reference collapses into an rvalue reference
  - All other cases collapse into an lvalue reference

&& Argument Parameters

- What is the effect of putting "&&" after an argument of specific type?

    func(Test&& x);

    - The parameter x is an rvalue reference to Test
    - It can only be bound to an rvalue (xvalue or prvalue)

&& Template Argument Parameters

- In a template function, what effect does putting "&&" after a template parameter argument have?

    template <typename T>
    func(T&& x);

    - This is a "forwarding reference"
    - It can be called with either an lvalue argument or an rvalue argument
    - The compiler will deduce the argument type and instantiate the function as appropriate

- Write a simple program to demonstrate that a function with a forwarding reference argument can be called with either an lvalue or an rvalue

Template Argument Deduction

- Consider this function with an argument which is a forwarding reference

        template <typename T>
        func(T&& x);

- What will be the prototype of the instantiated function if it is called with

  - An lvalue object of class Test

        func(Test& x);

  - An rvalue object of class Test

        func(Test&& x);

  - (For simplicity, assume that the instantiated function has the same name)

Why Are Forwarding References Useful?

- Explain why forwarding references are useful
  - When we want to overload a function on the value type of its argument
  - If the functions are otherwise identical, we can use a forwarding reference for the argument
  - The compiler will instantiate the overloads as needed