

Perfect Forwarding Solutions

- Write a trivial class X
- Write three overloaded functions g() which take an instance of X as argument by lvalue reference, const lvalue reference and rvalue reference respectively
- Each version of g() prints out the type of the argument

- Write a generic function `f()` which takes a forwarding reference to `X` and forwards it to `g()` directly
- Write a program which calls `f()` with an argument which is
 - An lvalue
 - A const
 - An rvalue reference
- Observe which versions of `g()` are called
- Explain the results

Output (direct)

Lvalue

Modifiable version of g // OK

Const lvalue

Immutable version of g // OK

Rvalue reference

Modifiable version of g // Wrong! x is passed as an lvalue

Output (std::move)

Lvalue

Moveable version of g // Wrong! x is passed as an rvalue reference

Const lvalue

Immutable version of g // OK

Rvalue reference

Moveable version of g // OK

Output (std::forward)

Lvalue

Modifiable version of g // OK

Const lvalue

Immutable version of g // OK

Rvalue reference

Moveable version of g // OK

- What is the purpose of `std::forward()`?
 - It performs a cast to rvalue reference, unless its argument is an lvalue reference
- Why was `std::forward()` needed when the language already had `std::move()`?
 - `std::move` performs an unconditional cast, which prevents perfect forwarding
- If a variable is passed to `std::forward()`, is it safe to use its data again afterwards?
 - No, because it may have been moved from