

Perfect Forwarding Exercises

- 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

- Rewrite your function `f()` so the argument passed to `g()`
 - is wrapped inside a call to `std::move()`
 - is wrapped inside a call to `std::forward()`
- What differences do you observe from the direct version?
- Explain the results

- What is the purpose of `std::forward()`?
- Why was `std::forward()` needed when the language already had `std::move()`?
- If a variable is passed to `std::forward()`, is it safe to use its data again afterwards?