Annoyances

Mechanical repetition of expressions might be required

As mentioned in *Use Cases — Creating an auxiliary variable of generic type* on page 28, using **decltype** to capture ~~a value~~ of an expression that is about to be used or ~~for the return value of an expression~~ can often lead to repeating the same expression in multiple places, three distinct ~~ones~~ in the earlier example.

An alternate solution to this problem is to capture the result of the **decltype** expression in a **typedef**, **using** type alias, or as a defaulted template parameter, but such an approach runs into the problem that it can be used only after the expression is valid. A defaulted **template** parameter cannot reference parameter names because it is written before them, and a type alias cannot be introduced prior to the return type being needed. A solution to this problem lies in using Standard Library function `std::declval` to create expressions of the appropriate type without needing to reference the actual function parameters by name:

```
template <typename A, typename B,
          typename Result = decltype(std::declval<const A&>() +
                                     std::declval<const B&>())>
Result loggedSum(const A& a, const B& b)
{
    Result result = a + b;  // no duplication of the decltype expression
    LOG_TRACE << a << " + " << b << "=" << result;
    return result;
}
```

Here, `std::declval`, a function that cannot be executed at run time and is only appropriate for use in **unevaluated contexts**, produces an expression of the specified type. When mixed with **decltype**, `std::declval` lets us determine the result types for expressions without needing to or even being able to construct objects of the needed types.

See Also

- "**using** Aliases" (§1.1, p. 133) explains that often it is useful to give a name to the type yielded by **decltype**, which is done with a **using** alias.

- "**auto** Variables" (§2.1, p. 195) illustrates how **auto** variables have a similar but distinct type deduction to that computed by **decltype**.

- "*Rvalue* References" (§2.1, p. 710) describes value categories that can be obtained for arbitrary expressions using **decltype**.