Despite the defaulted copy constructor, `Connection` will not be copy-constructible as `std::unique_ptr` is a noncopyable type. Some compilers *might* produce a warning on the declaration of `Connection(const Connection&)`, but they are not required to do so since the example code on the previous page is well formed and would produce a compilation failure only if an attempt were made to default-construct or copy a `Connection`.[4]

If desired, a possible way to ensure that a defaulted special member function has indeed been generated is to use **`static_assert`** (see Section 1.1."**`static_assert`**" on page 115) in conjunction with an appropriate trait from the `<type_traits>` header:

```
class IdCollection
{
    std::vector<int> d_ids;

public:
    IdCollection() = default;
    IdCollection(const IdCollection&) = default;
    // ...
};

static_assert(std::is_default_constructible<IdCollection>::value,
              "IdCollection must be default constructible.");

static_assert(std::is_copy_constructible<IdCollection>::value,
              "IdCollection must be copy constructible.");

// ...
```

Routinely using such compile-time testing techniques can help to ensure that a type will continue to behave as expected at no additional runtime cost, even when member and base types evolve as a result of ongoing software maintenance.

See Also

- "Deleted Functions" (§1.1, p. 53) describes a companion feature, `= delete`, that can be used to suppress access to implicitly generated special member functions.
- "**`static_assert`**" (§1.1, p. 115) describes a facility that can be used to verify at compile time that ~~undesirable~~ copy and move operations are declared to be accessible.
- "*Rvalue* References" (§2.1, p. 710) provides the basis for move operations, namely, the move-constructor and move-assignment special member functions, which too can be defaulted.

[4]Clang 8.0.0 (c. 2019) and later produces a diagnostic with no warning flags specified. MSVC 12.0 (c. 2013) produces a diagnostic if `/Wall` is specified. As of this writing, GCC 12.1 (c. 2021) produces no warning, even with both `-Wall` and `-Wextra` enabled.