Each of the constructs introduced by the keyword **`template`** within the `my` namespace in the previous example represents a separate explicit-instantiation definition. These constructs instruct the compiler to generate object-level definitions for general templates declared in `my_vector.h` specialized on the built-in type **`int`**. Explicit instantiation of individual member functions, such as `length()` in the example, is, however, only rarely useful; see *Annoyances — All members of an explicitly defined template class must be valid* on page 374.

Having installed the necessary explicit-instantiation definitions in the component's `my_vector.cpp` file, we must now go back to its `my_vector.h` file and, without altering any of the previously existing lines of code, *add* the corresponding explicit-instantiation declarations to suppress redundant local code generation:

```
// my_vector.h:
#ifndef INCLUDED_MY_VECTOR  // internal include guard
#define INCLUDED_MY_VECTOR

namespace my  // namespace for all entities defined within this component
{

// ...
// ...  everything that was in the original my namespace
// ...

                    // -----------------------------------
                    // explicit-instantiation declarations
                    // -----------------------------------

extern template class Vector<int>;
    // Suppress object code for this class template specialized for int.

extern template std::size_t Vector<double>::length() const;  // BAD IDEA
    // Suppress object code for this member, only specialized for double.

extern template void swap(Vector<int>& lhs, Vector<int>& rhs);
    // Suppress object code for this free function specialized for int.

extern template std::size_t vectorSize<int>;  // C++14
    // Suppress object code for this variable template specialized for int.

extern template std::size_t Vector<int>::s_count;
    // Suppress object code for this static member definition w.r.t. int.

}  // close namespace my

#endif  // close internal include guard
```

Each of the constructs that begins with **`extern template`** in the example above are explicit-instantiation declarations, which serve only to suppress the generation of any object code