2. A **free-function** template, `swap`, that operates on objects of corresponding specialized `Vector` type

3. A **const** C++14 variable template, `vectorSize`, that represents the number of **bytes** in the **footprint** of an object of the corresponding specialized `Vector` type

Any use of these templates by a client might and typically will trigger the depositing of equivalent definitions as object code in the client translation unit's resulting `.o` file, irrespective of whether the definition being used winds up getting inlined.

To eliminate object code for specializations of **entities** in the `my_vector` component, we must first decide where the unique definitions will go; see *Annoyances — No good place to put definitions for unrelated classes* on page 373. In this specific case, we own the component that requires specialization, and the specialization is for a ubiquitous built-in type; hence, the natural place to generate the specialized definitions is in a `.cpp` file corresponding to the component's header:

```
// my_vector.cpp:
#include <my_vector.h>  // We always include the component's own header first.
    // By including this header file, we have introduced the general template
    // definitions for each of the explicit-instantiation declarations below.

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

template class Vector<int>;
    // Generate object code for all nontemplate member functions and definitions
    // of static data members of template my::Vector having int elements.

template std::size_t Vector<double>::length() const;  // BAD IDEA
    // In addition, we could generate object code for just a particular member
    // function definition of my::Vector (e.g., length) for some other
    // argument type (e.g., double).

template void swap(Vector<int>& lhs, Vector<int>& rhs);
    // Generate object code for the full specialization of the swap free-
    // function template that operates on objects of type my::Vector<int>.

template const std::size_t vectorSize<int>;  // C++14 variable template
    // Generate the object-code-level definition for the specialization of the
    // C++14 variable template instantiated for built-in type int.

template std::size_t Vector<int>::s_count;
    // Generate the object-code-level definition for the specialization of the
    // static data member of Vector instantiated for built-in type int.

}  // close namespace my
```