```
{
    bool result = false;  // nonintersecting until proven otherwise
    // ...
    return result;
}

}  // Close lib namespace

#endif  // INCLUDED_LIB_INTERVAL


// lib_interval.cpp:
#include <lib_interval.h>
```

This library component above defines, in the namespace `lib`, an implementation of (1) a class template, `Interval`, and (2) a function template, `intersect`.

Let's also consider a trivial application that uses this library component:

```
// app.cpp:
#include <lib_interval.h>  // Include the library component's header file.

int main(int argv, const char** argc)
{
    lib::Interval<double> a(0, 5);  // instantiate with double type argument
    lib::Interval<double> b(3, 8);  // instantiate with double type argument
    lib::Interval<int>    c(4, 9);  // instantiate with int    type argument

    if (lib::intersect(a, b))  // instantiate deducing double type argument
    {
        return 0;  // Return "success" as (0.0, 5.0) does intersect (3.0, 8.0).
    }

    return 1;  // Return "failure" status as function apparently doesn't work.
}
```

The purpose of this application is merely to exhibit a couple of instantiations of the library *class* template, `lib::Interval`, for type arguments **`int`** and **`double`**, and of the library *function* template, `lib::intersect`, for just **`double`**.

Next, we compile the application and library translation units, `app.cpp` and `lib_interval.cpp`, and inspect the symbols in their respective corresponding object files, `app.o` and `lib_interval.o`:

```
$ gcc -I. -c app.cpp lib_interval.cpp
$ nm -C app.o lib_interval.o

app.o:
0000000000000000 W lib::Interval<double>::Interval(double const&, double const&)
```