```
                // Load result from null-terminated input; return 0 (on
                // success) or nonzero (with no effect on result).
        };

        template <typename T>
        double analyze(const Parser<T>& parser);
    }

    inline namespace v2    // Notice that use of inline keyword has moved here.
    {
        template <typename T>
        class Parser
        {
            // ...

        public:  // Note incompatible change to Parser's essential API.
            Parser();
            int parse(T* result, const char* input, std::size_t size);
                // Load result from input of specified size; return 0
                // on success) or nonzero (with no effect on result).
        };

        template <typename T>
        double analyze(const Parser<T>& parser);
    }
}
```

When we release this new version with `v2` made **`inline`**, all existing clients that rely on the version supported directly in `parselib` will, by design, break when they recompile. At that point, each client will have two options. The first one is to upgrade the code immediately by passing in the size of the input string (e.g., `23`) along with the address of its first character:

```
void client()
{
    // ...
    int status = parser.parse(&result, "...( MyClass value )...", 23);
    // ...                                                    ^^^^ Look here!
}
```

The second option is to change all references to `parselib` to refer to the original version in `v1` explicitly:

```
namespace parselib
{
    namespace v1  // specializations moved to nested namespace
    {
```