Recommendations for Specifying "Hidden Friends"

Document #: WG21 P1601R0
Date: 2019–03–10
Audience: Authors/reviewers of standard library wording
Reply to: Walter E. Brown <webrown.cpp@gmail.com>
Daniel Sunderland <dsunder@sandia.gov>

Contents

Abstract

This paper proposes wording to be used by proposal authors who wish to specify a non-member function as a *hidden friend*.

Everything is made of one hidden stuff.
— Ralph Waldo Emerson

Good merchandise, even hidden, soon finds buyers.
— Plautus

1 Introduction

The net effect of granting **`friend`**ship seems reasonably well-understood among C++ programmers. Rather less well-known, it seems, are some of the mechanical details and their implication with respect to name lookup.

When first declared via a **`friend`** declaration, the befriended entity's name (if unqualified) is injected into the nearest enclosing namespace. This is reasonable, as the named entity is not a member of the class granting friendship and so must become a member of some namespace. However, such name injection does not implicitly make that name visible to qualified or unqualified lookup; only argument-dependent lookup can find such an otherwise hidden name.[1,2]

When there is no additional, out-of-class/namespace-scope, declaration of the befriended entity, such an entity has become known as a *hidden friend* of the class granting friendship. Were there such an out-of-class/namespace-scope declaration, the entity would be no longer hidden, as the second declaration would make the name visible to qualified and to unqualified lookup.

There have been recent discussions[3] about employing this hidden friend technique, where applicable, throughout the standard library so that the declared entities (typically operator

[1] Per [N4800:basic.lookup.argdep]/1: "When the *postfix-expression* in a function call is an *unqualified-id*, other namespaces not considered during the usual unqualified lookup may be searched, and in those namespaces, namespace-scope friend function or function template declarations not otherwise visible may be found" (cross-references elided).

[2] Per [N4800:class.friend]/7: "A friend function defined in a class is in the (lexical) scope of the class in which it is defined. A friend function defined outside the class is not" (cross-references elided).

[3] E.g., those held in LEWG during the Kona 2019 WG21 meeting.

functions such as the new spaceship operator) would be found via ADL only. Because the library has not previously deliberately restricted lookup in this way, there is no precedent for specifying such a requirement. The remainder of this paper provides specification guidance to proposal authors who intend to impose such a requirement.

2 Proposed guidance

We recommend the following two-part approach for use each time an author intends to specify a hidden friend requirement on a function or function template:

1. The befriended entity should be fully defined within the class synopsis. (Brace-enclosed *see below* may be used when the definition is too bulky or is otherwise inconvenient to be specified *in situ*.)
2. In addition, the following sentence should become part of the entity's specification via an accompanying *Remarks:* element: "This function is to be found via argument-dependent lookup only."[4]

By way of example, consider the following class (namespace qualification omitted for clarity):

```
#include <ostream>
#include <compare>

class C  {
  friend ostream& operator << ( ostream&, C const& )  { see below }
  friend auto operator <=>( C const&, C const& )  = default;
};
```

Unless made unnecessary by pre-existing blanket wordng to the same effect, the additional specification that accompanies each of the above befriended operator functions would provide a *Remarks:* element that incorporates the sentence recommended above.

Finally, note that the example fully defines both **`friend`** functions inline, i.e., within the definition of the class.[5] This is important, because an out-of-line definition would also constitute a namespace-scope declaration. As pointed out above, any such declaration or redeclaration enables the usual qualified and unqualified name lookup and would thus defeat the intent of the hidden friend technique.

3 Bibliography

[N4800] Richard Smith: "Working Draft, Standard for Programming Language C++." ISO/IEC JTC1/SC22/WG21 document N4800 (pre-Kona mailing), 2019–01–21. https://wg21.link/n4800.

4 Document history

Rev.	Date	Changes
0	2019–03–10	• Published as P1601R0, post-Kona mailing.

[4] Once might envision that a future draft of the C++ standard would include blanket library wording that may make such per-function specification unnecessary. Until then, we recommend that each befriended function be accompanied by the above-recommended wording restricting successful lookup of befriended functions to ADL only.

[5] Recall that **`= default`** is a valid *function-body*, and thus qualifies as a function definition. See [dcl.fct.def.general]/1.