Proposal to Fix Local Class Friend Declarations

(X3J16/96-0187 = WG21/N1005)
October 29, 1996

R. Michael Anderson, Edison Design Group

The Problem

The current version of the working paper includes changes (to 3.3.1 [basic.scope.pdecl] and 7.3.1.2 [namespace.memdef]) that have affected how the following program is to be interpreted:

```
void f() {
  class A {
    friend class B;   // ::B vs. local B (pre-Stockholm)
    enum E { e };
  };
  class B {
    int f() {
      return A::e;    // access error vs. okay (pre-Stockholm)
    }
  };
}
```

Here is what the current version of 7.3.1.2 says:

> If a friend declaration first declares a class or function, and the name of the class or function is unqualified, the friend class or function is a member of the innermost enclosing namespace.

This replaces language in 3.3.1 that has now been removed:

> A class declared as a friend with a declaration of the form:
>
> friend class-key identifier ;
>
> and not previously declared is introduced into the smallest enclosing non-class scope that contains the friend declaration.

For non-local classes the "innermost enclosing namespace" scope is the same as the "smallest enclosing non-class scope", but that is not the case for local classes.

Discussion

It appears (based on private mail I've exchanged with Bill Gibbons and others) that the new interpretation of friend class declarations within local classes is an unintended side-effect of the changes to 3.3.1 and 7.3.1.2.

Although one might argue that the status quo is acceptable, I prefer to regard this as a bug to be fixed. However, whatever fix is proposed, I think we need to remain consistent with the new model introduced by the Stockholm change:

> Friend declarations that are the initial declaration of a class or function introduce an entity "invisibly" into an enclosing scope.
>
> The enclosing scope into which the entity is "invisibly" added is also the outermost scope in which lookup occurs to decide whether this is the initial declaration.

In other words, both name lookup and "the invisible name trick" need to be taken into account, and this will inevitably affect consistency with the pre-Stockholm draft in certain cases.

For example:

```
  class X;
  namespace N {
    class A {
      friend class X;
      friend class Y;
    };
  }
  void f() {
    class B {
      friend class X;
      friend class Y;
    };
  }
```

  Pre-Stockholm:

    In class N::A friend declarations cause X and Y to be injected into
      namespace A.
    In local class B friend declarations find ::X and cause local class Y
      to be injected into the scope of function f.

  Post-Stockholm:

    In class N::A friend declarations cause X and Y to be added invisibly
      to namespace A.
    In local class B friend declarations find ::X and cause class Y to
      be added invisibly to the global scope.

  Making local and non-local classes work the same:

    In class N::A friend declarations cause X and Y to be added invisibly
      to namespace A.
    In local class B friend declarations cause X and Y to be added
      invisibly to the scope of function f.

Three Options

In private email several of us have examined a couple of options, each of which fixes the bug identified above:

  (1) disallowing friend declarations in local classes:

```
      void f() {
        class X { };
        class B {
          friend class X;    // Not allowed
        };
      }
```

  (2) requiring friend declarations of local classes to match an
      existing visible declaration in the same block:

```
      class X;
      void f() {
        class Z;
```

```
        class B {
          friend class X;    // Error: no class X found
          friend class Y;    // Error: no class Y found
          friend class Z;    // Okay
        };
      }
```

(3) extending the friend lookup rules and "invisible name trick" to include local scopes as well as namespace scopes:

```
      class X;
      void f() {
        class B {
          friend class X;    // Adds local class X to function scope
          friend class Y;    // Adds local class Y to function scope
        };
      }
```

Option (1) is simplest to add to the working paper (and would probably have little effect on real-world programs, since access control in local classes is not a very useful concept); however, it has been discussed and rejected before.

Option (2) is also relatively simple to add to the WP. However, it has the (slight) disadvantage of making some previously valid programs ill-formed.

Option (3) introduces strict consistency between friend declarations in local and non-local classes, but it is more difficult to describe in the working paper. Moreover, I suspect that some previously valid programs remain valid but would have different semantics.

Working Paper Changes

Here are the Working Paper changes that will be required:

For all options, the following change should be made to 7.3.1.2 [namespace.memdef]:

-- Add to the second sentence of paragraph 3 the phrase "in a non-local class", so that it reads:

> If a friend declaration in a non-local class first declares a class or function, and the name of the class or function is unqualified, the friend class or function is a member of the innermost enclosing namespace.

For option (1) (to disallow friend declarations in local classes):

-- add to paragraph 2 of 11.4 [class.friend]:

> A friend declaration shall not appear in a local class [class.local].

-- change the first sentence of paragraph 6 of 11.4 to

> A function can be defined in a friend declaration if and only if the function name is unqualified.

For option (2) (to require a prior declaration):

-- add to paragraph 2 of 11.4:

If a friend declaration appears in a local class [class.local], the name shall either be a qualified name or refer to a class or function previously declared in the innermost enclosing non-class scope. [Example:

```
class X;
void a();
void f() {
  class Y;
  extern void b();
  class A {
    friend class X;    // error
    friend class Y;
    friend class Z;    // error
    friend void a();   // error
    friend void b();
    friend void c();   // error
  };
}
```

--end example]

For option (3) (to describe how friend declarations add invisible entities to the innermost enclosing non-class scope):

-- add to paragraph 2 of 11.4:

If a friend declaration appears in a local class [class.local] and the name specified is an unqualified name, a prior declaration is looked up without considering scopes outside the innermost enclosing non-class scope. If there is no prior declaration, the friend class or function belongs to innermost enclosing non-class scope, but the name of the friend is not found by simple name lookup in the innermost enclosing non-class scope until a matching declaration is provided in that scope.

Recommendation

I would personnaly prefer that we adopt option (1), but it is probably too substantive a change at this point.

Therefore, I recommend adoption of option (2). It is easy both to grasp and to specify, and it answers all practical concerns. It also fixes a bug and is therefore an appropriate change at this point in the process.

Option (3), which at first seems attractive, is harder to specify because of the always ugly interaction between friend function declarations and block extern declarations. (I'm not completely satisfied with the text I proposed above; not only is it unclear what it means for a function declaration to "belong" to a block scope, but I'm afraid there may be lookup issues I've missed.)