Named Requirements for C++0X Concepts, version 2

Document #: WG21/N2780 = J16/08-0290
Date: 2008-09-18
Revises: N2581
Project: Programming Language C++
Reference: ISO/IEC IS 14882:2003(E)
Reply to: Walter E. Brown <wb@fnal.gov>
Chris Jefferson <chris@bubblescope.net>
Alisdair Meredith <alisdair.meredith@codegear.com>
James Widman <widman@gimpel.com>

Contents

Reality: What a concept!

— ROBIN WILLIAMS

1 Introduction

This paper updates the proposed wording of N2581. It deletes one section as requested by the Evolution Working Group, and incorporates all changes requested by the Core Working Group.

2 Proposed wording

This section's proposed wording is with respect to N2741. This proposal is purely an extension to N2741; except for very small additions to the underlying grammar, it requires no changes to any existing wording.

In 14.9.3 [concept.refine], augment the grammar definition of *requirement-specifier*; in 14.10.1 [temp.req], augment the grammar definition of *requirement* and add a grammar definition of *concept-instance-alias-def*. Text to be added is indicated in red:

refinement-specifier :

concept-instance-alias-def$_{opt}$ ::$_{opt}$ *nested-name-specifier*$_{opt}$ *concept-id*

requirement :

concept-instance-alias-def$_{opt}$ ::$_{opt}$ *nested-name-specifier*$_{opt}$ *concept-id*
! ::$_{opt}$ *nested-name-specifier*$_{opt}$ *concept-id*

concept-instance-alias-def :

identifier =

Append the following new paragraph to 3.3.1 [basic.scope.pdecl]:

The point of declaration for the *identifier* in a *concept-instance-alias-def* is immediately after the *concept-id* of its *requirement* or *refinement-specifier*.

Append the following after 14.10.1 [temp.req] p5 ("A negative requirement requires..."):

A *concept-instance-alias-def* defines its *identifier* to be an alias of the concept instance given in its *requirement* or *refinement-specifier*. When the *concept-instance-alias-def* appears in a *member-requirement* (9.2), the potential scope of the *identifier* begins at its point of declaration and terminates at the end of the constrained member's declaration. When the *concept-instance-alias-def* appears in the optional *requires-clause* of an *axiom-definition* (14.9.1.4), the potential scope of the *identifier* begins at its point of declaration and terminates at the end of the *axiom-definition*. Otherwise, a *concept-instance-alias-def* inserts the *identifier* as a name in the scope of:

- the template parameters of the concept, when the *concept-instance-alias-def* appears in a *refinement-specifier* (14.9.3);
- the enclosing concept, when the *concept-instance-alias-def* appears in an *associated-requirement* (14.9.1.3); or
- the template parameters declared in the *template-parameter-list* immediately before the `requires` keyword, when the *concept-instance-alias-def* appears in the optional *requires-clause* of a *template-declaration*.

[*Example:*

```
concept A<typename X, typename Y, typename Z> {
  typename result_type;
}

concept B<typename X, typename Y> {
  typename result_type;
}

concept C<typename T> {
  typename R;
}

template<typename T>
  requires J = C<T>
```

```
J::R f( T );
  // qualified lookup finds type name R within the concept C (3.4.3.3)

auto concept D<typename Op, typename Elem> {
  requires a = A<Op, Elem, Elem>;
  requires B<a::result_type, Elem>;
  typename result_type = a::result_type;
};
```

—*end example*]

If a *concept-instance-alias-def* appears in a *requirement* that is the pattern of a pack expansion, the program is ill-formed. [*Example:*

```
concept C<typename ... Ts> {}

template<typename ... Ts>
  requires a = C<Ts>...  // error:  requirement aliases may refer only
                         // to requirements that are not pack expansions
void
  f( Ts... );
```

—*end example*]

3 Acknowledgments

We thank the Fermi National Accelerator Laboratory's Computing Division, sponsor of Fermilab's participation in the C++ standards effort, for its past and continuing support of efforts to improve C++ for all our user communities.