Core issues 743 and 950: Additional `decltype(...)` uses

Notes

The wording changes proposed in this paper address national body comment JP 8 (Core issue 743) to allow `decltype(...)` as a name qualifier. In addition, they also address Core issue 950 (allowing `decltype(...)` as a `base-specifier`) and the CWG's decision to allow the construct when forming destructor calls. For consistency's sake, the proposed wording also enabled `decltype(...)` for `mem-initializer-ids` and pseudo-destructor calls.

I made an attempt to fold `decltype-specifier` into `class-name`, but that doesn't fit well with existing uses of that grammar term (which often assume that a `class-name` is indeed a "name"). In the end, I just modified the grammar terms for the specific constructs that are being augmented.

The changes are against N3000.

Wording Changes

In 3.4.3 [basic.lookup.qual] paragraph 1 change the first two sentences as follows:

```
The name of a class or namespace member or enumerator can be referred to after the `::` scope resolution operator (5.1) applied to a `nested-name-specifier` that nominates
its class, namespace, or enumeration. During the lookup for a name preceding the `::`
scope resolution operator, object, function, and enumerator names are ignored in a `nested-name-specifier` is not preceded by a `decltype-specifier`, lookup of the name
preceding that `::` considers only namespaces, types, and templates whose
specializations are types.
```

Add a production to the grammar rule for `unqualified-id` in the introduction of 5.1.1
[expr.prim.general] as follows:

```
unqualified-id:
   identifier
   operator-function-id
   conversion-function-id
   literal-operator-id
  :`~` `class-name`
   `~` `decltype-specifier`
   template-id
```
Change the indicated sentence in 5.1.1 [expr.prim.general] paragraph 6 as follows:
6 ... A class-name or decltype-specifier prefixed by ~ denotes a destructor; see 12.4.

Add a production to the grammar rule for nested-name-specifier in 5.1.1 [expr.prim.general] paragraph 6 as follows:
6 ...
    nested-name-specifier:
      type-name ::
      namespace-name ::
      decltype-specifier ::
      nested-name-specifier identifier ::
      nested-name-specifier template<opt simple-template-id> ::

Change the first sentence following this grammar rule as follows:
A nested-name-specifier that names denotes a class, optionally followed by the keyword template ...

In 5.1.1 [expr.prim.general] paragraph 6 insert the following sentence before the final note:
... The form ~ decltype-specifier also denotes the destructor, but it shall not be used as the unqualified-id in a qualified-id.

In 5.1.1 [expr.prim.general] paragraph 8 change the first sentence as follows:
8 A nested-name-specifier that names denotes an enumeration ...

In 5.2 [expr.post] paragraph 1, add the following production to the grammar rule for pseudo-destructor-name:
    pseudo-destructor-name:
      ...
      ~ decltype-specifier

In 5.2.4 [expr.pseudo] paragraph 1 change the first sentence as follows:
1 The use of a pseudo-destructor-name after a dot . or arrow -> operator represents the destructor for the non-class type named denoted by type-name or decltype-specifier.
In 5.3.1 [expr.unary.op] paragraph 10, change the following sentence as indicated:

There is an ambiguity in the unary-expression \( \neg x() \), where \( x \) is a class-name or decltype-specifier.

In 7.1.6.2 [dcl.type.simple] paragraph 1 replace the production

\[
\text{simple-type-specifier:}
\]

by

\[
\text{simple-type-specifier:}
\]

and add the following rule:

\[
\text{decltype-specifier:}
\]

In 8.3.1 [dcl.meaning] paragraph 1 insert the following sentence before the note:

The nested-name-specifier of a qualified declarator-id shall not begin with a decltype-specifier.

In 8.3.3 [dcl.mptr] paragraph 1 change the following phrase as indicated:

the nested-name-specifier names denotes a class

(one occurrence).

In 10 [class.derived] paragraph 1, replace the grammar rule for base-specifier:

\[
\text{base-specifier:}
\]

Core issues 743 and 950: Additional decltype(...) uses N3031=10-0021
Core issues 743 and 950: Additional `decltype(...)` uses

by

\[
\text{base-specifier:}
\]
\[
\begin{align*}
\text{base-type-specifier} & \text{ attribute-specifier}_{opt} \\
\text{virtual} & \text{ access-specifier}_{opt} \text{ base-type-specifier} \text{ attribute-specifier}_{opt} \\
\text{access-specifier} & \text{ virtual}_{opt} \text{ base-type-specifier} \text{ attribute-specifier}_{opt}
\end{align*}
\]

\[
\text{class-type-specifier:}
\]
\[
\begin{align*}
& \text{: :}_{opt} \text{ nested-name-specifier}_{opt} \text{ class-name} \\
& \text{decltype-specifier}
\end{align*}
\]

\[
\text{base-type-specifier:}
\]
\[
\text{class-type-specifier}
\]

In 10 [class.derived] paragraph 2, change the first sentence as follows:

2 The class-name in a base-specified type denoted by a base-type-specifier shall not be a class type that is not an incompletely defined class (Clause 9); this class is called a direct base class for the class being defined.

In 11.2 [class.access.base] paragraph 5 change the following phrase as indicated:

class named denoted by the nested-name-specifier (one occurrence).

In 11.5 [class.protected] paragraph 1 change the following phrase as indicated:

the nested-name-specifier shall named denote (one occurrence).

In 12.4 [class.dtor] paragraph 10, change the first sentence as follows:

10 In an explicit destructor call, the destructor name appears as a ~ followed by a type-name or decltype-specifier that names denotes the destructor’s class type.

In 12.6 [class.base.init] paragraph 1, change the grammar rule for mem-initializer-id as follows:
In 12.6 [class.base.init] paragraph 2, change the first sentence as follows:

2 Names in a *mem-initializer-id* (that do not appear in a *decltype-specifier* or a *template-argument-list*) are looked up in the scope of the constructor’s class and, ...

In 12.6 [class.base.init] paragraph 3, change the first sentence as follows:

3 A *mem-initializer-list* can initialize a base class using any *name class-type-specifier* that denotes that base class type.

In 12.6 [class.base.init] paragraph 6, change the first sentence as follows:

6 A *mem-initializer-list* can delegate to another constructor of the constructor’s class using any *name class-type-specifier* that denotes the constructor’s class itself.

In 12.6 [class.base.init] paragraph 7, change the following sentence as indicated:

A *mem-initializer* where the *mem-initializer-id* names denotes a virtual base class is ignored during execution of a constructor of any class that is not the most derived class.

In 12.6 [class.base.init] paragraph 8, change the first sentence as follows:

8 If a given non-static data member or base class is not named designated by a *mem-initializer-id* ...

In 12.6 [class.base.init] paragraph 10, change the first bullet as follows:

— First, and only for the constructor of the most derived class (1.8), virtual base classes are initialized in the order they appear on a depth-first left-to-right traversal of the directed acyclic graph of base classes, where “left-to-right” is the order of appearance of the base classes names in the derived class *base-specifier-list*. 
In 12.9 [class.inhctor] paragraph 8 change the following phrase as indicated:

the base class named denoted in the nested-name-specifier

(one occurrence).

In 14.7.2.4 [temp.dep.temp] change paragraph 4 as follows:

4. A template template-argument is dependent if it names a template-parameter or is a qualified-id with a nested-name-specifier which contains a class-name or a decltype-specifier that names denotes a dependent type.