1 The problem

Although sometimes prone to overuse, the using-directive construct is frequently important and valuable. From adding simple convenience, such as replacing the awkward

```cpp
std::cout << std::string("Hello ") + "world" << std::endl;
```

with the more natural

```cpp
using namespace std;

... cout << string("Hello ") + "world" << endl;
```

to simplifying organization-wide conventions, using-directives are a widely used part of the language.

Unfortunately, using-directives generally are not suitable for use in header files because they risk polluting the namespace of the source files that include the header files. What makes this especially bad is that the prevalence of templates in modern C++ programming means that much if not most code is in header files. Indeed, almost the entire Boost library consists of header files. Of course, it is possible to include such declarations on a method by method basis. However, the common best practice of keeping methods short makes this technique of limited value.

It is not an overstatement to say that the result of this has been that the using-directive construct has proved largely useless for me. As a result, most of my code (at least that in header files) looks like the awkward “Hello world” example above instead of the more natural version that uses namespace std.

2 Namespace Regions

In order to make using-directives better suited for use in header files as well as other delineated regions of code, we propose that it be possible to enclose regions of code in braces, similar to extern "C" declarations. For example,
using namespace std {
    class A { ... };
    class B {
        void foo() {
            cout << string("Hello ") + "world!" << endl;
        }
    };
}

This allows much greater control over the region in which using-directives are in effect. In particular, if one is writing a header file, they can enclose its body in the appropriate using-directive without risk of polluting the namespace of files that include that header.

As with extern "C" declarations (and gc_strict {...} declarations), a new scope is not created.

3 Implementation Status

A modified version of g++ 4.2.0 exists that implements this proposal. No technical issues of note arose.

4 Proposed Wording

In the beginning of Chapter 7, add the following BNF for compound-declaration:

```
compound-declaration:
    [declaration-seq_opt]
```

Change the beginning of Section 7.3.4 [namespace.udir] as follows:

```
7.3.4 Using directive [namespace.udir]
using-directive:
    using namespace ::opt nested-name-specifier_opt namespace-name ;
    using namespace ::opt nested-name-specifier_opt namespace-name compound-declaration
```

1. A using-directive shall not appear in class scope, but may appear in namespace scope or in block scope. [Note: when looking up a namespace-name in a using-directive, only namespace names are considered, see 3.4.6. — end note]

2. A using-directive that does not contain a compound-declaration specifies that the names in the nominated namespace can be used in the scope in which the using-directive appears after the using-directive. A using-directive that contains a compound-declaration specifies that the names in the nominated namespace can be used in the compound declaration. During unqualified name lookup (3.4.1), the names appear as if they were de-
clared in the nearest enclosing namespace which contains both the *using-directive* and the nominated namespace.