Scope of Variables Declared in for-init-statements

Ron Irvine, ron@sco.com,
SCO Canada,
5 June 1991.

The Problem

In document X3J16/91-0021, titled "Scope and Lifetime of Conditional-Constructed Objects", it was stated that the core language working group agreed to recommend that the Committee leave the definition of the scope of variables declared in for-init-statements as it appears in the C++ base document. There are a number of serious problems with scope of variables in a for statement that were not identified in the document.

Consider the following example:

```c
for (int i=0; i<3; ++i) {
    for (int i=0; i<4; ++i) {
        ;
    }
}
```

According to the May 14, 1991 draft proposal section 6.5.3 this is equivalent to:

```c
int i=0;
while (i<3) {
    int i=0;
    while (i<4) {
        ;
        ++i;
    }
    ++i; // note: this is the inner variable "i"
}
```

In fact the two constructions are NOT equivalent. While both compile without error, the second construction produces a loop that never completes. It is clear that the while statement equivalence of a for statement as specified in section 6.5.3 is in error.

A similar problem is the fact that:

```c
for (int i=0; i<3; ++i) {
    // statement
}
```

is NOT equivalent to:

```c
for (int i=0; i<3; ) {
    // statement
    ++i;
}
```
This is NOT an intuitive situation, consider:

```cpp
for (int i=0; i<3; ) {
    // lots of code ...
    for (int i=0; i<4; ++i) {
        // lots of code ...
        ++i;    // inner variable "i"
    }
}
```

The inner for statement could easily have been a macro expansion or pasted into the mid part of the outer for without the programmer being aware of the trailing ++i. Note that the outer loop never ends. The for-init-statement seems to give the programmer much more LOCAL control of their index variables (usually named "i"), but with the current rules this is clearly a deception.

Clearly the current definition of the for statement is broken. There are clear issues of ambiguity and error in the current specification.

**The Fix**

The correct solution is to limit the scope of a for-init-statement declaration to the for statement. The for statement would then be equivalent to:

```cpp
{
    for-init-statement
    while (expression-1) {
        statement
        expression-2;
    }
}
```

The committee should re-evaluate this issue. As shown above there are significant problems with the current definition of the for statement. It is broken and needs to be fixed. The current scope of the for-init-statement declaration is problematic and is non-intuitive.

It is understood that some existing code will need to be modified. However, the current practice of using an index variable after the loop exits can easily be fixed by hoisting the for-init-statement to before the for statement (still in the block enclosing the for statement). In fact, this change will clarify the original intent of the programmer (that the scope is limited to the enclosing block not just the for statement) and work for BOTH scoping rules.