Document Number: WG21/N2332=07-0192
Date: 2007-06-20
Reply to: Michael Spertus
mike_spertus@symantec.com

Argument Deduction for Constructors

Michael Spertus

1 Discussion

According to the standard, template argument deduction may be done for functions. However, many important C++ idioms, such as `for_each` loops and factory patterns, use class constructors as if they were functions. Unfortunately, template argument deduction is not done in that case. This makes such constructs substantially more difficult than they need to be, and also makes functors less function-like than they need to be.

For example,

```
using namespace std;
template<class OutputIterator>
class Foo() {
public:
  Foo(OutputIterator o) : os(o) {}
  void operator()(int i) { os << i*i << endl; }
private:
  OutputIterator os;
};

void bar(vector<int> vi) {
   for_each(vi.begin(), vi.end(), Foo(cout)); // Ill-formed!
}
```

Note that it is quite awkward to figure out the template argument for the Foo constructor. Although, `auto` could be used, it is as awkward here as it would be for functions, where template argument deduction is useful and desirable. A variation of the above example is given in Vandervoorde and Josuittis book, *Templates: The Complete Guide*.

We propose simply that template argument deduction is done for a constructor of a template class as if it were a template function with the same template parameters as its class, making the above legal.

2 Proposed Wording

In section 12.1 [class.ctor], insert a new paragraph 15 between the current paragraph 14 and 15.

> If the constructor for a template class is called in this way, template argument deduction (14.8.2) is performed as if the constructor were a template function with the same template parameters as the class.