Templates

Adrian Kosmaczewski

2008-03-13

Did you knew this is possible in C++? I didn't.

```
void Fun()
{
    class Local
    {
        //... member variables ...
        //... member functions ...
    };

    // ... code using Local ...
}
```

This feature is called "local classes" and is part of the standard; the limitations are that these local classes cannot have static member variables and cannot access nonstatic local variables. But, as Alexandrescu points out (page 28), you can use them in template functions:

```
class Interface
{
public:
    virtual void Fun() = 0;
    // ...
};

template <class T, class P>
Interface* makeAdapter(const T& obj, const P& arg)
{
    class Local : public Interface
    {
    public:
        Local(const T& obj, const P& arg)
            : obj_(obj), arg_(arg) {}

        virtual void Fun()
```

```
        {
            obj_.Call(arg_);
        }

    private:
        T obj_;
        P arg_;
    };

    return new Local(obj, arg);
}
```

Not only that, but partial template specialization and template-based compile-time verifications just blew me away. The second chapter of the book was realizing I just haven't had the slightest clue about all the cool things you could do with C++!