Subject: Re: using Ctrl::Add; required for templates / overloaded virtual functions
Posted by mrjt on Wed, 23 Jun 2010 14:38:18 GMT
View Forum Message <> Reply to Message

I don't understand.

Option 2 is correct - ArrayCtrl doesn't overload Add(Ctrl &) (it's not even virtual), so obviously it's being called on the Ctrl base class.

Option 1 doesn't work because ArrayCtrl doesn't overload Add(Ctrl &), so the pointer signature is incorrect.

If you add the following to MyD then both options work correctly:

```
class MyD
 : public ArrayCtrl
{
public:
 typedef MyD CLASSNAME;
 virtual ~MyD() {}

 void    Add(Ctrl& ctrl) { Ctrl::Add(ctrl); Update(); }
};
```

Option 1: Calls MyD::Add

```
void (MyD::* mfp)(Ctrl &) = &MyD::Add;
(md.*mfp)(l);
```

Option2 calls Ctrl::Add

```
void (Ctrl::* mfp2)(Ctrl &) = &Ctrl::Add;
(md.*mfp2)(l);
```

This is also possible, and perhaps most useful:

```
void (MyD::* mfp)(Ctrl &) = &Ctrl::Add;
```

(Tested with latest SVN and MSC8)