Subject: answers to some questions
Date: Mon, 9 Mar 92 10:20:26 EST

Some questions came up yesterday dealing with C++. I'll answer them here with the attached code example.

First, on Lloyd's question of how to write a ''logically'' const member function: one that should be allowed to modify some part of its data. C++ strictly does not allow logical const-ness; see the examples on page 177 of the ARM or on pages 231-233 of Lippman's C++ Primer (2nd ed.), where it is explained that the type of 'this' in a const member function of class X is 'const X* const' (constant pointer to a constant object of type X). To get around this you can forcibly cast away the const-ness of 'this' (see comment 1 in the class defn. of Inner).

The other question came from Mark Holdaway; he found different behavior between passing an inner object vs. passing an explicit reference to the object, when passing the argument to a function that expected a reference:

```
        void foo( Inner& ) ;                // expects a reference to an Inner
        Inner& iref = outobj.innerobj ;     // explicit reference

        foo( outobj.innerobj ) ;
vs.
        foo( iref ) ;
```

Should these yield the same result? Yes. The code below demonstrates this.

```
    #include <iostream.h>

    class Inner {
     public:
      int itag ;
      int v[20] ;
      Inner( int ) ;

      //  comment 1
      //  Allow Add10 to modify the data member 'itag', even
      //  though Add10 is a const member function.

      void Add10() const { ((Inner* const)this)->itag += 10 ; } ;
      void Print() const ;
    } ;

    Inner::Inner(int arg) : itag(arg)
    {
      cout << "Inner: ctor entered with itag =" << itag << endl ;
    }

    void
    Inner::Print() const
    {
      cout << "   Inner::Print itag = " << itag << endl ;
    }

    class Outer {
     public:
      int otag ;
      Inner innerobj ;
      Outer(int) ;
      void Print() const ;
    } ;

    Outer::Outer(int arg) : otag(arg), innerobj(arg*2)
    {
      cout << "Outer: ctor entered, otag=" << otag << endl ;
    }

    void
    Outer::Print() const
    {
      cout << "   Outer::Print otag = " << otag << endl ;
    innerobj.Print() ;
  }

  void
  foo( const Inner& iarg )
  {
    cout << "   foo-- iarg is at address: " << (long*) &iarg << endl ;
    cout << "   foo-- invoking Add10 on iarg" << endl ;
    iarg.Add10() ;
    cout << "   foo-- invoking Print on iarg" << endl ;
    iarg.Print() ;
  }

  int
  main(int, char**)
  {
    Outer outobj(111) ;
    cout << "Entered main!\n" << endl ;

    // About to do stuff with Inner now...
    Inner& iref = outobj.innerobj ;
    cout << "outobj.innerobj is at address " << &outobj.innerobj << endl ;
    cout << "iref is at address " << &iref << "\n" << endl ;
    cout << "1. invoking Print on outobj (before actions)" << endl ;
    outobj.Print() ;

    // About to call foo...
    cout << "call foo with arg outobj.innerobj..." << endl ;
    foo( outobj.innerobj ) ;
    cout << "2. invoking Print on outobj (after 1st call)" << endl ;
    outobj.Print() ;

    cout << "call foo with arg iref..." << endl ;
    foo( iref ) ;
    cout << "3. invoking Print on outobj (after 2nd call)" << endl ;
    outobj.Print() ;

    cout << "\nExiting main!" << endl ;
  }
```

And here is the output produced:

```
  Inner: ctor entered with itag =222
  Outer: ctor entered, otag=111
  Entered main!

  outobj.innerobj is at address 0xf7fffad4
  iref is at address 0xf7fffad4

  1. invoking Print on outobj (before actions)
     Outer::Print otag = 111
     Inner::Print itag = 222
  call foo with arg outobj.innerobj...
     foo-- iarg is at address: 0xf7fffad4
     foo-- invoking Add10 on iarg
     foo-- invoking Print on iarg
     Inner::Print itag = 232
  2. invoking Print on outobj (after 1st call)
     Outer::Print otag = 111
     Inner::Print itag = 232
  call foo with arg iref...
     foo-- iarg is at address: 0xf7fffad4
     foo-- invoking Add10 on iarg
     foo-- invoking Print on iarg
     Inner::Print itag = 242
  3. invoking Print on outobj (after 2nd call)
     Outer::Print otag = 111
     Inner::Print itag = 242

  Exiting main!
```

Andrew Klein