Automatic Casts redux

Asked 16 years, 3 months ago Modified 16 years, 3 months ago Viewed 146 times

0

After I messed up the description of my previous post on this I have sat down and tried to convey my exact intent.

I have a class called P which performs some distinct purpose. I also have PW which perform some distinct purpose on P. PW has no member variables, just member functions.

From this description you would assume that the code would follow like this:

```
class P
{
public:
    void a( );
};

class PW
{
public:
    PW( const P& p ) : p( p ) { }
    void b( );
    P& p;
};

class C
{
public:
    P GetP( ) const { return p; }
private:
    P p;
};

// ...
    PW& p = c.GetP( ); // valid
// ...
```

However that brings up a problem. I can't call the functions of P without indirection everywhere.

```
// ...
    p->p->a( )
// ...
```

What I would like to do is call p->a( ) and have it automatically determine that I would like to call the member function of P.

Also having a member of PW called *a* doesn't really scale - what if I add (or remove) another function to P - this will need to be added (or removed) to PW.

c++ casting

Share
Improve this question
Follow

edited May 23, 2017 at 10:27
Community Bot
1 ● 1

asked Sep 24, 2008 at 9:56
graham.reeds
16.5k ● 17 ● 75 ● 138

Again you are returning a copy of 'p' from the class C with GetP(). Is this what you want? You should probably return a reference. – Loki Astari Sep 24, 2008 at 15:20

3 Answers

Sorted by: Highest score (default)

2

You could try overriding operator* and operator-> to return access to the embedded p. Something like this might do the trick :

```
class P
{
  public:
    void a( ) { std::cout << "a" << std::endl; }
};

class PW
{
  public:
    PW(P& p) : p(p) { }

    void b( ) { std::cout << "b" << std::endl; }

    P & operator*()  { return p;  }
    P * operator->() { return &p; }

  private:
    P & p;
};

class C
{
  public:
    P & getP() { return p; }

  private:
    P p;
};


int main()
{
  C c;
```

```
    PW pw(c.getP());

    (*pw).a();
    pw->a();
    pw.b();

    return EXIT_SUCCESS;
}
```

This code prints

```
a
a
b
```

However, this method may confuse the user since the semantic of operator* and operator-> becomes a little messed up.

Share
Improve this answer
Follow

edited Sep 24, 2008 at 11:44

answered Sep 24, 2008 at 10:46
Luc Touraille
81.9k ● 16 ● 99 ● 139

That's precisely what I had in mind. I can't remember overloading the * and -> operator but my mind plays tricks on me at times... – graham.reeds Sep 24, 2008 at 11:40

1

If you make P a superclass to PW, like you did in your previous question, you could call p->a() and it would direct to class P. It seems like you've already considered and rejected this though, judging from this question. If so, would you care to elaborate why this wont work for you?

Share Improve this answer Follow

answered Sep 24, 2008 at 10:05
Rik
29.2k ● 14 ● 51 ● 69

Your response just made something click in my brain. Thanks!! My initial way was almost right just missing the spark - which you provided. – graham.reeds Sep 24, 2008 at 10:12

0

I think you need to think through what kind of relationship exists between PW and P.

Is it an is-a relationship? Are instances of PW instances of P? Then it would make sense to have PW inherit from P.

Is it a has-a relationship? Then you should stick with containment, and put up with the syntactic inconvenience.

Incidentally, it's generally not a good idea to expose a non-const reference to a member variable in a class's public interface.

Share Improve this answer Follow

answered Sep 24, 2008 at 15:00

JohnMcG
8,795 ● 6 ● 43 ● 49