Can I alias a member of a base class in a derived class?

Asked 9 years, 5 months ago Modified 9 years, 5 months ago Viewed 3k times

10

2

Say I have the following classes:

```
template <class T>
class Base {
  protected:
    T theT;
    // ...
};

class Derived : protected Base <int>, protected Base <float> {
  protected:
    // ...
    using theInt = Base<int>::theT;     // How do I accomplish this??
    using theFloat = Base<float>::theT; // How do I accomplish this??
};
```

In my derived class, I would like to refer to `Base::theT` using a more intuitive name that makes more sense in the Derived class. I am using GCC 4.7, which has pretty good coverage of C++ 11 features. Is there a way of using a `using` statement to accomplish this kind of how I tried in my example above? I know that in C++11, the `using` keyword can be used to alias types as well as eg. bring protected base class members into the public scope. Is there any similar mechanism for aliasing a member?

c++ c++11

Share Edit Follow Flag

edited Dec 9, 2012 at 5:16
Praetorian
103k ● 17 ● 231 ● 318

asked Dec 9, 2012 at 4:44
Nicu Stiurca
8,319 ● 6 ● 38 ● 46

4 I think you either need references or probably rather a function which won't take up space in the derived class. :| – Xeo Dec 9, 2012 at 4:49

1 Answer

Sorted by: Highest score (default)

8

Xeo's tip worked. If you are using C++ 11, you can declare the aliases like so:

```
int   &theInt   = Base<int>::theT;
float &theFloat = Base<float>::theT;
```

模板就是类的共同基础,
所以模板的数据成员 可以看做 模板特例化后的类的静态成员变量,
所以得用::访问

If you don't have C++11, I think you can also initialize them in the constructor:

```
int   &theInt;
float &theFloat;
// ...
Derived() : theInt(Base<int>::theT), theFloat(Base<float>::theT) {
  theInt = // some default
  theFloat = // some default
}
```

EDIT: The slight annoyance is that you can't initialize the the value of those aliased members until the main body of the constructor (ie, inside the curly braces).

Share Edit Follow Flag

answered Dec 9, 2012 at 5:07
Nicu Stiurca
8,319 ● 6 ● 38 ● 46

4 Note that this increases the size of the derived class by `sizeof(void*)` times the number of references. That's why I included the suggestion of a simple getter function that is named `theXXX`. – Xeo Dec 9, 2012 at 5:53

Yes, I suppose you are right. Luckily, I don't think an extra 8 bytes will kill me since I don't have a lot instances of the Derived class, so I can stick with the easier-to-type reference version when I access the data member. – Nicu Stiurca Dec 9, 2012 at 7:47