How to understand such "two consecutive templates" in c++ by using a mimic minimum example?

Asked today Active today Viewed 42 times

1

I only understan some simple template usage in C++.

Recently I met the following code snippet from some OpenFOAM code, and it confues me for weeks long.

(1) Could you please help me by giving a minimum working example to explain such "two consecutive templates" usage?

(2) Can I just replace the two tempalte with single template, i.e., `template<Type, Type2>`

1

Oh, please ignore the unknown classes such as Foam, fvMatrix.

Thanks~

```
template<class Type>
template<class Type2>
void Foam::fvMatrix<Type>::addToInternalField
(
    const labelUList& addr,
    const Field<Type2>& pf,
    Field<Type2>& intf
) const
{
    if (addr.size() != pf.size())
    {
        FatalErrorInFunction
            << "addressing (" << addr.size()
            << ") and field (" << pf.size() << ") are different sizes" << endl
            << abort(FatalError);
    }

    forAll(addr, facei)
    {
        intf[addr[facei]] += pf[facei];//intf是diag
    }
}
```

c++ templates

Share Edit Delete Flag

edited 14 hours ago

asked 16 hours ago
pengfei_guo
197 ● 7

4 Can you show the in class definition of `Foam::fvMatrix` it's probably a templated class with a templated member function. – Richard Critten 16 hours ago

@RichardCritten Thanks for your attention :) OpenFOAM is open source, and `Foam::fvMatrix` can be find here openfoam.com/documentation/guides/latest/api/.... However, I just lost the way when I reading the document. – pengfei_guo 16 hours ago

@ Richard Critten, I am sorry if my quesion is silly. I searched the book "C++ template a complete guide", but I didn't find an answer. As you mentioned, if a templated class with a templated member function, I am still not very clear about such usage. Why use two tempaltes, instead of a single template with two template parameters? – pengfei_guo 16 hours ago

Because the class template and the member function templates depend on different types. If the class was defined as `template< class T1, class T2 >` every time the member function used a different `T2` it would instantiate a whole new class. – Richard Critten 15 hours ago

1 Answer

Active Oldest Votes

1

It's when you define a *member template* in a *class template*.

A common example will be a `copy assignment operator` for a template class. Consider the code

```
template <typename T>
class Foo {                    函数声明. 没有函数体{}
    // Foo& operator= (const Foo&); // can only be assigned from the current
specilization

    template <typename U>
    Foo& operator= (const Foo<U>&); // can be assigned from any other specilizations
};

// definition                  函数实现. 铛铛铛! 连着的template终于现身
template <typename T>
template <typename U>
Foo<T>& Foo<T>::operator= (const Foo<U>&) {
  用T实例化Foo                          用U实例化Foo
}
```

For copy assignments from different specializations, you have to do this. The first `template <typename T>` belongs to the class template `Foo`, and the second `template <typename U>` belongs to the copy assignment operator, it's an inner template.

For your second question, the answer is **No**. One template parameter list can only introduce one template. There are two templates here. The second class template `Foam::fvMatrix` is of no business with template parameter `T2`. (For those who are interested in source code, see header and implementation)

Aside: this topic is covered in the 5.5.1 section of *C++ Templates: The Complete Guide*.

Share Edit Following Flag

answered 12 hours ago
Nimrod
856 ● 5 ● 15

Very clear and insightful example! Thanks for your great help! Have a nice day:) – pengfei_guo 34 mins ago

glad it helps. Have a nice day too. – Nimrod 20 mins ago