成员模板函数不能为虚函数，同时也不能有默认参数

1．不能为虚函数的原因：
编译器在编译一个类的时候，需要确定这个类的虚函数表的大小。一般来说，如果一个类有N个虚函数，它的虚函数表的大小就是N，如果按字节算的话那么就是4*N。
如果允许一个成员模板函数为虚函数的话，因为我们可以为该成员模板函数实例化出很多不同的版本，也就是可以实例化出很多不同版本的虚函数，那么编译器为了确定类的虚函数表的大小，就必须要知道我们一共为该成员模板函数实例化了多少个不同版本的虚函数。显然编译器需要查找所有的代码文件，才能够知道到底有几个虚函数，这对于多文件的项目来说，代价是非常高的，所以才规定成员模板函数不能够为虚函数。
举个例子：

```
class Calc
{
public:
template virtual T Add(const T& lhs,const T& rhs)
{
T sum = lhs + rhs;
return sum;
}
};

void main()
{
Calc ins;
int nResult = ins.Add(1,2);
double fResult = ins.Add(1.0,2.0);
}
```

那么编译器需要查看main的代码，才能够知道类Calc一共有两个虚函数，一个是virtual int Add(const int&,const int&)，另一个是virtual double Add(const double&,const double&)。
因为只有这样才能够确定该类的虚函数表的大小为2，虽然可行，但是费事，代价高。

2．不能够有默认参数的原因：
这里所说的默认参数有两种，必须分清楚，第一种是函数的默认参数，第二种是函数模板的默认模板参数。
成员模板函数和普通函数一样，可以有函数的默认参数（第一种），
但是不可以有函数模板的默认模板参数（第二种）。如：

```
//第一种的情况：
template <typename T>
T sum(T* b,T* e,T result = T()) //这里的result就是函数的默认参数，允许
{
while(b!=e)
result += *b++;
return result;
};
//第二种的情况：
template <typename T = char>//这里的char为默认模板参数，不允许
T f(T a, T b)
{
return a + b;
};
```

之所以不允许有函数模板的默认模板参数，是因为函数模板的模板参数是在调用的时候编译器根据实参的类型来确定的，如对于上面的第二种情况，

```
int result = f(1,2) //int f(int a,int b)，确定T为int
double result = f(1.0,2.0) //double f(double a,duoble b))，确定T为double
```

那么template中的T的缺省值char就毫无意义了，也毫无必要。

但注意，对于上面那些可以通过函数模板的实参演绎模板参数类型的情况，指定缺省的模板实参确实没有意义和必要。但某些情况下，有一些模板参数(比如作为函数的返回值类型)无法通过函数的实参演绎来确定，这个时候指定缺省的模板实参就很有必要，如下情况：

```
// 返回类型RT无法通过实参演绎获得，这个时候指定RT的缺省实参就是非常必要的
template <typename RT, typename T1, typename T2>
inline RT add(T1 a, T2 b)
{
    return a + b;
}
```

【转自】http://blog.chinaunix.net/uid-20781394-id-516009.html

分类: C++ Template

小天_y
关注 - 63
粉丝 - 96
+加关注

« 上一篇： 模板中的名字查找问题
» 下一篇： 帮 C/C++ 程序员彻底了解链接器