|  |  |
| --- | --- |
| 1. 类模板 2. 定义类模板 3. 实例化类模板 4. 在模板作用域中引用模板类型 5. 类模板的成员函数 6. Check和元素访问成员 7. Blob构造函数 8. 类模板成员函数的实例化 9. 在类代码内简化模板类名的使用 10. 在类模板外使用类模板名 11. 类模板和友元 12. 一对一友好关系 13. 通用和特定的模板友好关系 14. 令模板自己的类型参数称为友元 15. 模板类型别名 16. 类模板的static成员 | 2.template <typename T> class Blob  3.显式模板实参  4.shared\_ptr<vector<T>> data  5.template <typename T> ret-type Blob<T>::member-name(parm-list)  8.Blob<string> articles = {“a”,”asf”,”d”}  9.blobptr& operator++();  11.friend class BlobPtr<T>  Friend bool operator==<T>()..  12.friend class Pal<C>  Template <typename T> friend class pal2;  14.template <typename type> class bar{  Friend type;}  15.typedef blob<string> strblob;  Template<typename T> using twin = pair<T,T>  Twin<string> authors;  Template <typename T> using partNo = pair<T,unsigned>  partNo<string> books;  16.static std::size\_t ctr;  Template <typename T>  Size\_t Foo<T>::ctr = 0;  Foo<int> fi;  Auto ct = Foo<int>::count();  Ct = fi.count();  Ct = foo::count();//error |
| 1. 类模板 2. 类模板是用来生成类的蓝图的。与函数模板不同的是，编译器不能为类模板推断模板参数类型。（就是必须得传入类型，不能是值）   Vector<string>，必须传入string类型这个模板实参列表来代替模板参数的  但是后面又能传入值  非类型模板参数的类模板，可以传递值来直接给非类型模板参数  但是  类型模板参数的类模板，不可以通过值来推断类型给模板参数   1. 定义类模板 2. 如上，template开始，后跟模板参数列表。类模板定义中，将模板参数当做替身，代替使用模板时用户需要提供的类型或值 3. 实例化类模板 4. 额外信息是：显式模板实参列表，绑定到模板参数。 5. 当指定模板实参后，编译器会实例化出特定类，会重新blob模板，将模板参数T的每个实例替换为给定的模板实参 6. Note：一个类模板的每个实例都形成一个独立的类。类型blob<string>与blob<int>等其它类型没有任何关联 7. 在模板作用域中引用模板类型 8. 类模板的名字不是一个类型名。 9. 此shader\_ptr指向一个保存类型为T的对象的vector实例。 10. 类模板的成员函数 11. 可以在类模板内部，也可以在类模板外部定义成员函数，在内部的隐式为内联函数 12. 类模板的每个实例都有其自己版本的成员函数。因为具有与类模板相同的模板参数，所以需要template开始 13. Check和元素访问成员 14. 。。 15. Blob构造函数 16. 构造函数也要以模板参数开始   Template <typename T>  Blob<T>::Blob()::…   1. 类模板成员函数的实例化 2. 默认下，类模板的成员函数只有当程序用到它时才进行实例化。没有使用，就不实例化，使得即使某种类型不能完全符合模板操作的要求，我们仍然能用该类型实例化类   Note：对于实例化了的类模板，其成员只有在使用时才被实例化   1. 在类代码内简化模板类名的使用 2. 当编译器处理模板自身引用时就好像我们已经提供了与模板参数匹配的实参一样   BlobPtr<T>& operator++();   1. 在类模板外使用类模板名 2. 在类模板外定义其成员时，遇到类名才表示进入类的作用域，在里面的类不用重复模板实参。 3. 类模板和友元 4. 类与友元各自是否是模板相互无关的 5. 类模板包含一个非模板友元，被授权可以访问所有模板实例 6. 类模板包含一个模板友元，可以授权所有模板实例，也可以特定实例 7. 普通类包含一个模板友元，可以授权每个实例，也可以只授限定实例友元 8. 一对一友好关系 9. 通用和特定的模板友好关系 10. 必须先声明模板自身。   包含一个模板友元   1. 特定实例：   Friend class Pal<T>  所有模板实例  Template <typename X> friend class Pal2;//注意不要前置声明  包含一个非模板友元  friend class Pal3;   1. 令模板自己的类型参数称为友元 2. 友元通常是类或函数，但也可以用一个内置类型实例化，对内置类型的友好关系是允许的 3. 模板类型别名 4. 定义一个typedef引用实例化的类，不能定义一个typedef引用Blob<T> 5. 允许我们为类模板定义个类型别名 6. 一个模板类型别名是一族类的别名 7. 类模板的static成员 8. 与定义成模板的成员函数类似，将static数据成员也定义为模板 9. 必须有模板实例才能调用count，就变成只有一种方式调用了 | |