|  |  |
| --- | --- |
| 1. 定义模板 2. 函数模板 3. 实例化函数模板 4. 模板类型参数 5. 非类型模板参数 6. Inline和constexpr的函数模板 7. 编写类型无关的代码 8. 模板编译 9. 大多数编译错误在实例化期间报告 | 函数模板  模板参数列表  模板参数  Template <typename T>  Int compare(const T &v1, const T &v2)  模板实参  实例化  模板的实例  Compare(1,0)  Compare(vec1,vec2)  模板类型参数  Template<typename T, class U>  非类型参数  Template<unsigned N,unsigned M>  Int compare(const char (&p1)[N],const char (&p2)[m]  Template <typename T>inline T min();  if (less<T>()(v1, v2))return -1;  if (less<T>()(v2, v1))return 1; |
| 1. 定义模板 2. 像vector<>这种就是模板的使用，容器是具有泛型的定义，模板是泛型的基础 3. 一个模板是一个创建类或函数的蓝图或公式 4. 函数模板 5. 定义一个通用函数模板解决仅参数类型不一样重载的函数，这样就只需一个 6. 定义以关键字template开始，后跟一个模板参数列表，是一个逗号分隔的一个或多个模板参数 template <typename T,typename C>。。。用小于号和大于号包围起来的 7. 使用时，我们（隐式或显示的）指定模板实参，绑到模板参数上。 8. 每个模板参数都需要被推断或者显示的指出   20190609  就是函数列表得对应模板参数列表  不一定要对应，反正要推断出模板参数列表，模板参数分类型模板，非类型模板，都得推断出来。不管用没用到。  类型模板：在模板声明内被推断为类型  非类型模板：在模板声明内被推断为值，必须也要在函数中使用为形参的一部分才能推断出来   1. 实例化函数模板 2. 使用时，通过函数的实参推断模板参数的类型，就是传给函数形参的是值不是类型，则编译器会推断模板实参的类型再给模板参数。 3. 用推断出的模板参数为我们实例化一个特定版本的函数。就是，模板实参代替对应的模板参数创建出模板的一个新 实例 4. 编译器生成的版本通常被称为模板的实例 5. 模板类型参数   模板参数分：模板类型参数，模板非类型参数   1. T就是模板类型参数（是需要推断出来的类型），可以看做类型说明符，可以当做返回类型或函数的参数类型 2. 类型参数前必须使用关键字class或typename 3. 模板非类型参数 4. 非类型参数表示一个值而非一个类型，用特定类型名，而非typename或class 5. 模板被实例化时，非类型参数被一个用户提供的或编译器推断出的值所代替，非类型模板参数的模板实参必须是常量表达式 6. Copare(“hi”,”mom”);使用字面常量的大小代替N和M，从而实例化模板。 7. 非类型参数可以是一个整型、一个指向对象的函数类型的指针、左值引用。   绑定到非类型整型参数的实参必须是一个常量表达式  指针、引用非类型参数 的 实参 需要具有静态的生存期。或者nullptr或0给指针   1. 在模板定义内，模板非类型参数是一个常量值。在需要常量表达式的地方，可以使用非类型参数，如：指定数组大小。Unsigned N   Note：非类型模板参数的模板实参必须是常量表达式 “mom”这个   1. Inline和constexpr的函数模板 2. 在模板参数列表之后，返回类型之前 3. 编写类型无关的代码 4. 原则：   模板中的函数参数是const的引用  函数体中的条件判断仅使用<  2） 第一条，保证函数可以用于不能拷贝的类型，用于处理大对象，运行速度也快  3) 第二条，因为这样保证函数处理的类型仅需支持<运算符，而不需要其它运算符  4) 若真心关心类型无关和可移植性，需要用less，因为less对于指针提供良好定义。  Note:模板程序应该尽量减少对实参类型的要求   1. 模板编译 2. 编译器遇到模板定义时，并不生成代码，只有实例化时，即使用模板时才生成代码 3. 定义类时，可以在头文件声明并定义类，但函数可以只是声明在另一个文件定义函数，   而模板不同：模板的头文件包括声明与定义，  即：函数模板和类模板的成员函数需要在同一个头文件定义  关键概念：模板和头文件  模板包含两种名字：  不依赖模板参数的名字  依赖模板参数的名字，成员，函数  1.不管如何，在使用模板时，这两种都必须可见。  2.实例模板时，模板的定义，包括类模板的成员的定义，都必须是可见的。  3.用来实例化模板的所有函数、类型以及类型关联的运算符的声明都必须是可见的。就是调用者  4.应该提供一个头文件存放模板定义或类模板中的成员声明与定义。   1. 大多数编译错误在实例化期间报告 2. 第一阶段：检查语法错误 3. 第二阶段：遇到模板使用，检查实参数目是否正确 4. 第三阶段：才能发现类型相关的错误。依赖于编译器如何管理实例化，可能在链接时才报告。就是在将T转换为模板实参的类型在函数体中使用了这个类型并没有定义模板中的操作而会错误。   就是直至编译器在模板实参的类型给T并且使用函数时才会发现 | |