|  |  |
| --- | --- |
| 1. noexcept异常说明 2. 违反异常说明 3. 异常说明的实参 4. Noexcept运算符 5. 异常说明与指针、函数和拷贝控制 | 提供noexcept说明  不抛出说明  Noexcept(true)  Noexcept运算符  Noexcept(e)  Void (\*pf1)(int) noexcept = recoup;  Pf1 = recoup; |
| 1. **noexcept异常说明** 2. 预先知道某个函数不会抛出异常有好处 3. 特殊优化操作。简化调用该函数的代码 4. 没有Noexcept的代表不抛出说明 5. Noexcept需出现在声明与定义中，在尾置返回类型之前 6. 在typedef或类型别名中不能出现noexcept 7. 违反异常说明 8. 在声明不出现异常的函数体总抛出异常 9. 会抛出terminate 10. 所以noexcept用在：   a. 我们确认不会出现异常的函数  b. 我们根本不知道该如何处理异常所以直接结束程序   1. 异常说明的实参 2. noexcept(true)表示不会抛出 false表示可能 3. 一个可选的实参，必须能转换为bool 4. **Noexcept运算符** 5. 一个一元运算符，返回值是一个bool类型的右值常量表达式，不会求其对象的值 6. Noexcept(fun(i));.若fun不抛出异常为true，否则为faslse，是整个返回true 7. Noexcept（e），e调用的所有函数都做了不抛出声明且e本身不含有throw语句时，为true 8. Noexcept(noexcept(g())),可是是noexcept异常说明可是运算符 9. 异常说明与指针、函数和拷贝控制 10. 函数指针及该指针所指的函数必须具有一致的异常说明。 11. 不抛出异常函数指针对应不抛出异常 12. 可能抛出异常函数指针可以放不抛出异常和可能抛出异常 13. 虚函数若也定义了异常，则派生类的函数遵循以上规则 14. 编译器合成拷贝控制成员，也会有一个异常说明。若所有成员和基类的所有操作不会异常，则合成noexcept.任意一个抛出异常，那么就异常。 15. 若定义了一个析构函数但是没有为它提供异常说明，则编译器将合成一个异常说明。合成的异常说明将与假设由编译器为类合成析构函数时所得的异常说明一致。 | |