|  |  |
| --- | --- |
| 1. 捕获异常 2. 查找匹配的处理代码 3. 重新抛出 4. 捕获所有异常的处理代码 | Catch子句，异常声明  重新抛出  Catch(…) |
| 1. 捕获异常 2. Catch子句中的异常声明类似包含一个形参的函数形参列表 3. 声明的类型决定了处理代码能捕获的异常类型。必须是完全类型，左值引用，不能是右值引用 4. 若参数类型非引用是拷贝 5. Catch的参数是基类类型，可以用派生类对其初始化，若catch参数是非引用，则切掉一部分，是引用，则动态绑定   5）异常声明的静态类型将决定catch语句所能执行的操作。若catch的参数是基类，无法执行派生类的特有成员   1. 查找匹配的处理代码 2. 未必是最佳的匹配，是第一个与异常匹配的catch语句 3. 需派生类异常的处理代码在基类异常代码之前 4. 绝大数类型转换都不能被允许，除了一些细小的差别之外 5. 允许非常量向常量 6. 允许派生类和基类 7. 数组与指针，函数与函数指针 8. 算术类型 9. 重新抛出 10. 当前catch无法处理，可以在catch中重新抛出，将异常传递给另外一个catch，通过throw;不包含表达式，只能在catch相关的地方，不然报错 11. Catch语句改变了参数的内容再重新抛出，只有引用时才会抛出改变后的对象 12. 捕获所有异常的处理代码 13. 用省略号作为异常声明，称为：捕获所有异常catch(…) 14. 通常与throw重新抛出一起工作，既能单独出现，也能与其它几个catch语句一起出现，需要置底在最后位置，不然在其后的其它的catch语句不执行   小结:  try  {  throw new exception();  }  catch (exceptionType \*pet)  {  cout << "(exceptionType \*pet)" << endl;// 能捕获但是还是抛出，因为\*pet  }  Throw什么，捕获什么，即使不能处理写法也对的，可用…最后捕获\*pet | |