|  |  |
| --- | --- |
| 1. 拷贝、赋值与销毁 2. 拷贝构造函数 3. 合成拷贝构造函数 4. 拷贝初始化 5. 参数和返回值 6. 拷贝初始化的限制 7. 编译器可以绕过拷贝构造函数 | 拷贝构造函数、拷贝赋值运算符、移动构造函数、移动赋值运算符 析构函数。拷贝控制操作  合成拷贝构造函数  拷贝初始化  Explicit  Foo（const Foo&）;  String dot(10,’.’))//直接初始化  String s(dot);//直接初始化  String s3 = s2;//拷贝  String s4 = “999”;//拷贝  String s5 = string(100,’9’);拷贝  F(vector<int>(10)); |
| 1. 拷贝、赋值与销毁 2. 定义一个类时，显式或隐式地指定在此类型的对象拷贝、移动、赋值和销毁时做什么。通过五种特殊的成员函数来控制这些操作 3. 一个类没有定义的话，编译器会合成，但是对于一些类来说，使用合成的版本不好，并非我们所需 4. 拷贝构造函数   当用同类型的另一个对象初始化本对象时做什么，就是在初始化的时候，右侧对象可以先隐式转换对象再通过 = 赋给左侧运算对象   1. 拷贝构造函数的第一个参数必须是一个引用类型且任何额外参数都有默认值，并且不应该是explicit的因为需要隐式地被使用 2. 合成拷贝构造函数 3. 合成的版本是将参数的成员逐个拷贝到正在创建的对象中，非static成员。 4. 拷贝方式由每个成员的类型决定：类类型成员，使用拷贝构造函数，内置类型成员，直接拷贝。数组不能直接拷贝，合成版本会逐元素地拷贝一个数组类型的成员。如果数组元素是类类型，使用元素的拷贝构造来进行拷贝 5. 拷贝初始化 6. 直接初始化：使用普通的函数匹配选择匹配的构造函数 7. 拷贝初始化：将右侧运算对象拷贝到正在创建的对象中，如果需要的话要进行类型转换 8. 拷贝初始化不仅在 = 定义变量时发生，还有：   一个对象作为实参传给非引用类型的形参  从一个返回类型为非引用类型的函数返回一个对象  用花括号列表初始化一个数组中的元素或聚合类的成员   1. 某些类类型还会对他们所分配的对象使用拷贝初始化。如：容器使用insert或push容器会对其元素进行拷贝初始化。相对的，用emplace成员创建的元素都进行直接初始化。 2. 参数和返回值 3. 如上，非引用类型的参数要进行拷贝初始化，所以我们的拷贝构造函数的参数要是引用类型,会循环。 4. 拷贝初始化的限制 5. 使用了explicit的拷贝构造函数需要显示使用构造函数。 6. 编译器可以绕过拷贝构造函数   Test1 t2 = Test1("sdfsd");// 这个奇怪，直接构造对象,不会发生拷贝构造  Test1 t1("sdfsdf");// 也是直接构造对象，应该是直接使用函数，不会发生拷贝构造   1. Test1 t3 = string("sdfsd");//直接构造对象,不会发生拷贝构造   以上三个都会使用string的构造函数，不会发生拷贝构造，即使上面三个方式，只有  Test1 t4(t1);// 显示调用拷贝构造函数 接受一个t1对象  Test1 t5 = t2;// 隐式调用拷贝构造函数 | |