**C++学习笔记——对象构造和析构**

**1. 就地初始化**

自C++11起，类的非静态、非const成员变量，可以像许多主流编程语言那样，采用“就地初始化”方式，其效果等价于在构造函数的初始化列表中进行初始化。

文本

描述已自动生成

显然，就地初始化更加简明且易于理解，并且与许多主流编程语言（如Python、Java）语法相似。

**2. = delete**

C++11引入的新特性，仅可用于类的以下成员函数：构造函数、复制构造函数、移动构造函数、赋值运算符、移动赋值运算符。注意，不可用于析构函数，因为析构函数显然不能被删除。

作用：禁用默认函数（如类的复制构造和赋值成员函数）、防止意外重载、提升代码清晰性、提升报错准确性。

文本

描述已自动生成

注意：如果需要禁用虚函数，则应该沿用C++98已有的”= 0”声明，而不是”= delete”。例如：

virtual void func() = 0;

这样的虚函数被称为“纯虚函数”；包含纯虚函数的类称为抽象类，无法被实例化。

**3. = default**

C++11引入的新特性，仅可用于类的以下成员函数：无参构造函数、复制构造函数、移动构造函数、赋值运算符、移动赋值运算符、析构函数。

作用：声明函数默认行为，从而无需再手动实现函数体。

作用：比起手动实现默认函数体，= default声明更加简洁，并且编译效率更高。

**4. 引用成员变量**

C++允许以引用形式声明类的成员变量。

如果一个类包含引用成员变量（或者const成员变量），编译器会自动将这个类的默认赋值和移动赋值运算符设置为= delete，即禁用默认赋值行为。

如果确实要进行赋值，则必须手动重载其赋值和移动赋值运算符。如下图所示。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

引用成员变量的生命周期，比包含它的对象更长。这一机制极其违背直觉，很容易因此出现安全性漏洞。

目前来看，引用成员变量可能仅在lambda表达式中的引用捕获机制中有意义。总之，尽量不要将引用作为自定义类的成员变量。

**5. explicit**

仅可用于修饰构造函数、复制构造函数和移动构造函数，目的是在构造对象过程中，禁止隐式类型转换。

(1) 对于构造函数，绝大多数情况下，在构造函数调用过程中发生的隐式类型转换，都是预期之外的行为。因此，应尽量用explicit修饰构造函数，除非确实允许这种隐式类型转换发生。

(2) 如果复制/移动构造函数使用explicit修饰，那么如下使用’=’的复制/移动构造将无法通过编译。此时，只能采用带圆括号的复制构造方式。

(3) 但是，有些编译器会对调用explicit移动构造的代码进行优化，实际上并不调用移动构造函数，以减少对象拷贝次数。这种情况下，即使移动构造函数用explicit修饰，代码中使用’=’进行移动构造，也能正常编译和运行。对于g++编译器而言，想要禁用这种优化，需要在编译时指定-fno-elide-constructors。

(4) 在开发实践中，为避免误解，也为了方便对象构造，不建议用explicit修饰复制/移动构造函数。

图片包含 文本

描述已自动生成

**6. enum class/struct**

C++11引入的语法，是对enum的扩展。enum class与enum struct完全等价，因此下文以enum class为例说明。

enum class用于需要限定对象取值范围的场景。例如：月份的有效取值范围为1到12之间的整数，此时就可以使用enum class。

相比C++98的enum，enum class禁用了与int类型之间的隐式类型转换，提升了安全性，这是其最大的优点。此外，enum class也具有更强的作用域限定。

**7. 重载类型转换运算符，实现从类到类内成员的隐式类型转换**

如果class B内包含class A成员变量，则class B可以重载类型转换运算符（它本身是B的成员函数），将B对象直接隐式转换为A对象。

如下代码，21行便是重载了类型转换运算符，允许将Font对象转换为其包含的FontHandle对象。有了21行的代码，33、36、37行的代码才能通过编译。

这是一个多数情况下**不推荐使用**的特性。具体见代码和注释。

