Week14

a. When should operator overloading be considered for a class?

应当极其谨慎地使用操作符重载，只有在语义非常清晰且合理的情况下才应考虑。

- **原则：** 操作符重载的使用应该尽量少(minimal)，并且只在非常有意义的情况下使用。如果其他开发者需要查看你的类定义才能明白该操作符的具体作用，那么这通常意味着设计是失败的。
- **适用场景：** 例如在定义数学相关的类(如 `Math` class 或 `Vector` 结构体)时，重载加号 `+` 或乘号 `*` 是非常有意义的，因为你可以直接写出 `a + b` 这样的代码，这比写 `a.Add(b)` 更简洁且可读性更强。
- **目的：** 主要是为了清理代码，使其更加精简(streamline)和易读。但切记不要为了重载而重载，以免产生糟糕的代码风格。

b. Where should the overloading of the output operator `<<` for a class be defined and why?

- **定义位置：** 必须在类(Class)的**外部**定义。
- **原因：** `<<` 操作符的左操作数通常是 `std::cout`(这是一个 `std::ostream` 对象)，而不是你的类对象本身。如果将其定义为成员函数，那么你的对象必须作为左操作数(即 `this` 指针)，这不符合 `std::cout << object` 的调用习惯。因此，我们需要在类外定义一个接受 `std::ostream` 和你的类对象作为参数的函数。

c. What is the difference between overloading a binary operator inside a class versus outside a class?

- **类内部(Inside)：** 当作为成员函数重载时(例如 `operator+`)，实际上只接受**一个参数**。左操作数隐式地为 `this` 指针(即调用该函数的对象实例)，右操作数作为参数传入。
- **类外部(Outside)：** 当作为非成员函数重载时(例如 `operator<<`)，它通常需要接受**两个参数**。第一个参数是左操作数(例如 `std::ostream`)，第二个参数是右操作数(你的类对象)。

d. How can a private member variable of a class be accessed by a non-member function?

可以通过将该非成员函数在类内部声明为 friend(友元) 来实现。

- 在视频示例中，为了让定义在类外部的 `operator<<` 能够访问 `String` 类的私有成员 `m_Buffer`，Cherno 将该操作符函数声明为该类的 `friend` 。这样，该非成员函数就可以直接访问类的私有数据(private members)了。

e. What is a shallow copy, and under what circumstances can it cause a program crash?

- **浅拷贝(Shallow Copy)：** 是指直接复制对象的所有成员变量的值。如果成员变量包含指针，它只复制指针的地址(即内存地址的数值)，而不复制指针所指向的实际内存数据。这会导致两个对象(原对象和副本)内部的指针指向同一个内存地址。

- **导致崩溃的情况：** 当涉及动态内存分配(如使用 `new`)时，浅拷贝会导致程序崩溃。
    - **原因：** 当这两个指向同一块内存的对象超出作用域(Scope)时，析构函数(Destructor)会被调用。第一个对象销毁时会 `delete` 这块内存。紧接着，第二个对象销毁时会试图再次 `delete` **同一块**已经被释放的内存。
    - 这种行为被称为 "Double Free"，会导致程序崩溃。

f. What is required to implement a deep copy?

为了实现深拷贝(Deep Copy)，你需要自定义一个拷贝构造函数(Copy Constructor)。

- **实现逻辑：** 在拷贝构造函数中，不能只复制指针值，而是需要：
    1. 根据源对象的大小，**分配(Allocate)** 一块新的独立的内存块(例如使用 `new char[...]`)。
    2. 将源对象指针指向的数据(实际内容)**复制(Copy)** 到这块新分配的内存中(例如使用 `memcpy` 或循环赋值)。
- 这样，新对象就拥有了自己独立的内存副本，修改或销毁它不会影响原对象。

g. What is the preferred way to pass an object into a function?

首选方式是通过 const 引用(const reference) 传递。

- **语法：** 例如 `const String& string` 。
- **原因：**
    1. **性能(Performance)：** 避免了不必要的拷贝(Unnecessary copying)和内存分配，这会显著提高程序运行速度。如果不使用引用，每次传参都会触发拷贝构造函数，进行完整的内存分配和数据复制。
    2. **安全(Safety)：** `const` 关键字保证了函数内部不能修改传入的原对象。