既然你嫌之前的太简单，那我们就直接把难度拉满。你要求综合考量 **The Cherno 前 40 集** 的内容，这意味着我们要在一个代码里同时处理：**面向对象 (OOP)、内存管理 (Stack vs Heap)、静态生命周期 (Static)、多态 (Polymorphism)、常量正确性 (Const Correctness) 以及 指针/引用 (Pointers/Refs)**。

这不再是填空题，这是一个微型系统的**架构设计题**。

### 终极挑战：硅谷芯片生命周期系统 (The Silicon Lifecycle System)

背景：

你正在为一家 IC 设计公司编写一个简单的仿真器。我们需要模拟不同的硬件组件（Component），追踪它们的数量，让它们工作，最后安全销毁它们。

**核心考点：**

* Ep 21: Static (追踪实例数量)
* Ep 26-29: Inheritance & Virtual Functions (多态与接口)
* Ep 31: Const (修饰函数与参数)
* Ep 35: Stack vs Heap (手动内存管理)
* Ep 39: The new keyword (实例化)
* **隐藏考点**：Virtual Destructor (内存泄漏陷阱)

### 需求规格说明书 (Spec Sheet)

你需要在一个 main.cpp 中实现以下所有类和逻辑：

#### 1. 基类 Component (抽象类)

* **成员变量**：
  + 一个 protected 的字符串 m\_Name（组件名称）。
  + 一个 **私有的静态 (static)** 整数 s\_Count，用来统计当前内存中有多少个 Component 存活。
* **构造函数**：
  + 接收名字，初始化 m\_Name。
  + 每次创建对象时，让 s\_Count 加 1。
  + 打印 "Constructed: [Name]"。
* **析构函数**：
  + **必须正确设计**以支持多态删除。
  + 每次销毁对象时，让 s\_Count 减 1。
  + 打印 "Destroyed: [Name]"。
* **成员函数**：
  + Process(): 一个**纯虚函数 (Pure Virtual)**。每个组件处理数据的方式不同。
  + GetName(): 返回组件名字。**要求**：必须保证该函数不会修改类成员，且返回引用以提高效率。
  + GetActiveCount(): 一个**静态 (static)** 函数，返回当前的 s\_Count。

#### 2. 派生类 CPU 和 GPU

* 继承自 Component。
* **实现 Process()**：
  + CPU 打印: "CPU [Name] is calculating data..."
  + GPU 打印: "GPU [Name] is rendering graphics..."

#### 3. 全局辅助函数

* **void Analyze(const Component& comp)**
  + 接收一个组件的**只读引用**。
  + 打印其名字，并调用其 Process() 方法。
* **void FactoryReset(Component\* comp)**
  + 接收一个组件的**指针**。
  + **安全检查**：如果是 nullptr，打印警告并返回。
  + 如果不是空，打印 "Resetting [Name]..."。

#### 4. main 函数 (内存战场)

逻辑必须严格按顺序执行：

1. 打印当前存活数量（预期：0）。
2. **在堆上 (Heap)** 创建一个包含 3 个 Component\* 指针的数组。
   * 指针 0 指向一个新的 CPU("Intel")。
   * 指针 1 指向一个新的 GPU("Nvidia")。
   * 指针 2 设为 nullptr（模拟故障插槽）。
3. 打印当前存活数量（预期：2）。
4. 使用 for 循环遍历这 3 个指针：
   * 如果指针有效，调用 Analyze 函数。
   * 如果指针有效，调用 FactoryReset 函数。
   * (注意：你的 FactoryReset 内部已经有 nullptr 检查了，但这里为了逻辑严密，怎么处理你自己决定)。
5. **核心内存清理**：
   * 使用 for 循环，将所有在堆上分配的对象**正确删除 (delete)**。
6. 打印当前存活数量（预期：0）。

### 你的任务

不要问我任何问题。不要问我“纯虚函数怎么写”。这都是前 40 集讲过的。

如果你的代码有以下任何一个问题，就算失败：

1. 内存泄漏（根据 s\_Count 验证）。
2. 多态调用失败（比如 Analyze 里只调用了基类的逻辑）。
3. const 位置放错。
4. 试图实例化抽象类。
5. 程序崩溃（未处理 nullptr）。

**这就不仅仅是写代码了，这是在用 C++ 搭建积木。给我看你的架构能力。**