### 🏛️ 试炼任务：SoC 固件管理器 (The SoC Firmware Manager)

背景：

你需要编写一个简单的 C++ 程序，模拟一个片上系统 (SoC) 的启动过程。这个 SoC 包含不同的硬件组件（CPU、内存、加速器），你需要用面向对象的方式管理它们。

**核心考点：**

* **Heap vs Stack** (Ep 38)
* **Virtual Functions & Polymorphism** (Ep 29)
* **Pure Virtual Functions (Interfaces)** (Ep 30)
* **Constructor & Member Initializer Lists** (Ep 26)
* **The explicit Keyword** (Ep 40)
* **Static** (Ep 21)
* **Destructors & Memory Cleaning** (Ep 24, 39)

### 🛠️ 详细需求 (Spec Sheet)

请在 VS Code 中创建一个 main.cpp，实现以下类和逻辑：

#### 1. 定义枚举 (Ep 23)

定义一个 enum 叫 Status，包含 STOPPED 和 RUNNING。

#### 2. 基类 Hardware (抽象类)

* **成员变量**：
  + m\_Name (字符串，硬件的名字)。
  + m\_Status (上面定义的枚举)。
* **静态成员 (Static)**：
  + s\_Count (统计当前有多少个硬件对象存在)。
* **构造函数**：
  + 接受 const char\* 或 std::string 作为名字。
  + **必须使用成员初始化列表 (Member Initializer List)** 初始化名字，并将状态设为 STOPPED。
  + 每次构造时，让 s\_Count 加 1。
* **虚析构函数 (关键考点)**：
  + 打印 "[Name] Destroyed"。
  + 让 s\_Count 减 1。
* **纯虚函数 (Pure Virtual)**：
  + virtual void Start() = 0; (每个硬件启动方式不同，强制子类实现)。
* **普通成员函数**：
  + void PrintStatus() const：打印当前名字和状态。**注意 const (Ep 34)**。

#### 3. 派生类 (Ep 27)

创建两个类继承自 Hardware：

* **类 CpuCore**：
  + Start() 实现：打印 "CPU [Name] is fetching instructions..."，并将状态改为 RUNNING。
* **类 NttAccelerator** (你的 DSA 加速器)：
  + Start() 实现：打印 "NTT [Name] is calculating fast..."，并将状态改为 RUNNING。

#### 4. 管理类 SoC (The Manager)

这是一个用来管理所有硬件的容器。

* **成员变量**：
  + Hardware\* m\_Devices[5] (一个存放**指针**的数组，最多存5个设备)。
  + int m\_DeviceCount (当前存了几个)。
* **构造函数**：
  + **使用 explicit 关键字 (Ep 40)**，防止隐式转换。
  + 初始化 m\_DeviceCount 为 0。
* **析构函数 (生死攸关)**：
  + **遍历数组，释放 (delete) 所有已添加的硬件指针。** (如果你忘了这一步，就是严重的内存泄漏)。
* **方法 AddDevice**：
  + 接受一个 Hardware\* 指针，把它存入数组。
* **方法 BootAll**：
  + 遍历数组，调用每个设备的 Start()。

#### 5. main 函数 (场景模拟)

1. **在栈 (Stack) 上** 创建一个 SoC 对象（例如叫 myChip）。
2. **在堆 (Heap) 上** 创建 1 个 CpuCore 和 1 个 NttAccelerator。
   * *注意：使用 new 关键字*。
3. 把这两个堆对象的指针 AddDevice 到 myChip 里。
4. 调用 myChip.BootAll()。
5. 打印当前的 Hardware::s\_Count (应该显示 2)。
6. main 函数结束。

### 💀 死亡判定标准 (Pass/Fail)

当你运行程序时，控制台必须**严格按顺序**输出以下内容。如果少了任何一行，视为**失败**。

**预期输出：**

Plaintext

CPU RiscyV is fetching instructions... <-- BootAll 调用的  
NTT AlgoUnit is calculating fast... <-- BootAll 调用的  
Active Hardware: 2 <-- s\_Count 打印的  
CPU RiscyV Destroyed <-- SoC 析构时自动触发的  
NTT AlgoUnit Destroyed <-- 同上

**陷阱提示（Consultant's Hint）：**

1. 如果最后两行 "...Destroyed" 没有出现，说明你的 SoC 析构函数没写对，或者**基类 Hardware 的析构函数忘了加 virtual**。这是面试中必挂的题。
2. 如果你能写出 SoC chip = 5; 这种代码且没报错，说明你忘了 explicit。

现在，关闭浏览器，打开 VS Code。

Show me the code.