根据你的《C++ 学习复习查漏补缺方案》，今天是 **Day 6**。

今日主题：数据完整性与 Const 契约 (Data Integrity & Const Contract)

FPGA 关联任务： 加密核心 (Crypto Core) —— 在硬件中，常数意味着“硬连线”的逻辑 (Hardwired Logic) 或 ROM，是绝对不可修改的铁律。

今天的训练核心是：**初始化列表的陷阱** 与 **Const 的物理意义**。

我为你设计了一个名为 **“时钟发生器的灾难” (The Clock Generator Disaster)** 的题目。这个题目看起来非常简单，但它包含了一个 90% 的 C++ 初学者（甚至老手）都会踩进去的**初始化顺序陷阱**。

请在 IDE 中创建项目，严格按照以下**规格说明书 (Spec Sheet)** 编写代码。

### 题目：不稳定的时钟源 (Unstable Clock Source)

背景：

你正在设计一个 FPGA 内部的时钟管理模块。你需要根据“基频”和“倍频系数”计算出最终的输出频率。为了保证效率，硬件架构师要求你必须使用 C++ 的 成员初始化列表 (Member Initializer List) 来初始化所有变量。

#### 步骤 1：定义类 ClockGenerator

请严格按照以下顺序在 private 区域定义成员变量（顺序不可变，这是硬件寄存器的物理布局）：

C++

class ClockGenerator  
{  
private:  
 // 寄存器布局（物理顺序）  
 int m\_OutputFreq; // 最终输出频率  
 int m\_Multiplier; // 倍频系数  
 const int m\_BaseFreq; // 基频（硬件固定，必须是 const）  
   
public:  
 // ... 构造函数 ...  
 int GetOutput() const { return m\_OutputFreq; }  
};

#### 步骤 2：编写构造函数

* **输入参数**：int base (基频), int mult (倍频)。
* **要求**：
  1. **必须**使用成员初始化列表初始化所有三个变量 (m\_OutputFreq, m\_Multiplier, m\_BaseFreq)。不要在 {} 函数体里写赋值语句。
  2. 在初始化列表中，试图按照“逻辑顺序”写：先把 m\_BaseFreq 和 m\_Multiplier 初始化好，最后再算 m\_OutputFreq。
  3. **计算公式**：m\_OutputFreq = base \* m\_Multiplier; (或者是 m\_BaseFreq \* m\_Multiplier)。

#### 步骤 3：编写 Main 函数

* 实例化一个对象：ClockGenerator clk(100, 5); （基频 100Hz，5 倍频）。
* 打印 clk.GetOutput()。
* **预期输出**：应该是 500。

### 你的任务

1. 编写并运行代码。
2. **关键观测**：你的输出是 500 吗？还是一个奇怪的负数/巨大的随机数？
3. **核心拷问**：
   * 如果输出了垃圾值，为什么？（提示：请检查 private 成员变量的**声明顺序**与**初始化列表的执行顺序**有什么关系？）
   * 为什么 m\_BaseFreq 必须在初始化列表中初始化，不能在函数体 {} 里写 m\_BaseFreq = base;？

**请提交你的代码截图和运行结果，我们来解剖这个“未定义行为”。**