这是为你生成的 **Day 8: DSP Signal Saturator (数字信号饱和运算器)** 工程开发规格说明书。

请严格按照以下命名规范和功能逻辑进行实现。

### 1. 命名规范 (Naming Standard)

为了保证代码的可读性和工程规范，请强制使用以下命名：

* **类名 (Class Name)**: Signal
* **私有成员变量 (Private Member)**: int m\_Value (用于存储信号数值)
* **函数参数名 (Argument)**:
  + 构造函数输入: value
  + 操作符重载输入: other (例如 s1 + s2，s2 就是 other)

### 2. 功能逻辑规格 (Functional Specifications)

#### **A. 构造函数 (Constructor)**

* **函数原型**: Signal(int value)
* **业务逻辑**: 输入清洗（Input Sanitization）。
  + 必须将输入 value 限制在 0 到 255 之间。
  + 如果 < 0，m\_Value 设为 0。
  + 如果 > 255，m\_Value 设为 255。
  + 否则，m\_Value 等于 value。

#### **B. 加法操作符 (Operator +)**

* **函数原型**: Signal operator+(const Signal& other) const
* **业务逻辑**: **不可变饱和加法**。
  1. 计算 this->m\_Value 与 other.m\_Value 的和。
  2. 如果和大于 255，结果必须被截断为 255（Saturation）。
  3. **禁止修改**参与运算的两个对象（即 this 和 other 都不能变）。
  4. **必须返回**包含计算结果的一个新 Signal 对象。

#### **C. 累加操作符 (Operator +=)**

* **函数原型**: Signal& operator+=(const Signal& other)
* **业务逻辑**: **可变链式累加**。
  1. 将 other.m\_Value 加到当前对象 (this->m\_Value) 上。
  2. 同样执行饱和逻辑（上限 255）。
  3. **关键要求**: 必须返回当前对象本身的引用（\*this），以支持 (s1 += s2) += s3 这种链式写法。

#### **D. 输出流操作符 (Operator <<)**

* **函数原型**: friend std::ostream& operator<<(std::ostream& stream, const Signal& s)
* **业务逻辑**: **格式化打印**。
  1. 这是一个全局函数（非成员函数），需要在类中声明为 friend。
  2. 向 stream 输出格式为 [SIG: 数值] 的字符串。
  3. 必须返回 stream 引用，以支持 cout << s1 << endl。

### 3. 验收测试用例 (Acceptance Test)

请在 main 函数中复现以下场景，确保输出与预期一致：

1. **测试加法 (+)**:
   * Signal s1(200); Signal s2(100);
   * Signal sum = s1 + s2;
   * **预期**: sum 内部值为 255，且 s1 仍为 200。
2. **测试累加 (+=)**:
   * s1 += s2;
   * **预期**: s1 内部值变为 255。
3. **测试链式调用**:
   * Signal a(10); Signal b(20); Signal c(30);
   * (a += b) += c;
   * **预期**: a 最终值为 60。

**Action**: 请在 IDE 中创建项目并开始编码。