C++ 讲义：`if` 条件嵌套

一、判断中的再判断

当一个判断逻辑过于复杂，无法通过单一层级的 `if-else if-else` 有效解决时，需要引入一种更强大的结构——**条件嵌套**。

- **定义**：**在一个 `if`、`else if` 或 `else` 的 `{}` 代码块内部，再放入另一个完整的 `if` 结构**，这就构成了条件嵌套。
- **核心思想**：它映射了人类解决复杂问题时的"分步决策"的思维模式。**外层 `if` 负责确定一个大的前提或类别，而内层 `if` 则在此外层前提下，处理更细分的子问题。**

通过嵌套，我们可以将一个复杂的大问题变成了一系列更简单的小问题。

二、语法结构与执行流程

2.1 基本语法

条件嵌套的本质，就是将一个 `if` 语句块作为另一个 `if` 语句块的执行内容。

```
if (外层条件) {
    // 外层条件成立后，进入此代码块
    if (内层条件) {
        // 当 外层条件 和 内层条件 同时成立时，执行这里的代码
    } else {
        // 当 外层条件 成立，但 内层条件 不成立时，执行这里的代码
    }

} else {  // 注意这里的 else 匹配的是 if (外层条件)
    // 当 外层条件 不成立时，执行此代码块
    // 这里也可以继续嵌套...
}
```

2.2 执行流程解析

计算机在执行嵌套结构时，遵循"**由外到内，层层把关**"的原则：

1. **判断外层条件**：首先，只看最外层的 `if` 条件。
2. **决定是否进入**：
    - 如果外层条件为 `false`，程序将直接跳过整个外层 `{}` 代码块，不会去检查内层的 `if` 条件。
    - 如果外层条件为 `true`，程序才会进入外层代码块中，并开始判断内层的 `if` 条件。
3. **判断内层条件**：进入内层后，判断内层的 `if-else` 结构，根据条件的真假来决定执行哪个子分支。

三、基础用法与实例

3.1 场景一：会员等级与消费金额折扣

需求：一个商场打折规则如下：

- 必须是会员才能享受折扣，并且会员如果消费满 500 元，打 8 折；如果消费不满 500 元，打 9 折。
- 非会员不打折。

分析：

- **外层判断**：首要前提是“是否为会员？”
- **内层判断**：在“是会员”这个大前提下，再细分“消费金额是否满500？”的条件。

代码实现：

```
bool is_member = true;
double total_cost = 600.0;

if (is_member == true) {
    // 代码能进入此区域，说明此人一定是会员
    cout << "尊贵的会员您好" << endl;

    if (total_cost >= 500) {
        // 会员，且消费满500
        cout << "享受8折优惠！" << endl;
    } else {
        // 会员，但消费未满500
        cout << "享受9折优惠！" << endl;
    }

} else {
    // 代码没有进入此区域，说明此人一定不是会员
    cout << "抱歉，非会员不参与打折活动。" << endl;
}
```

3.2 场景二：三数排序 (书本例3-5)

任务：输入三个整数 `a, b, c`，将它们从大到小排序输出。

分析：

- **外层判断**：先确定三个数中最大的那个数是谁（三选一）。
- **内层判断**：在最大数确定的前提下，只需比较剩下两个数的大小（二选一）。

代码实现参见书本例3-5，本处略。

四、总结

1. **嵌套构建复杂逻辑**：当单一维度的判断不足以解决问题时，使用 `if` 嵌套结构来处理“层层递进”的判断。
2. **分治思想的体现**：条件嵌套是一种“分而治之”的思想。外层 `if` 解决主要矛盾，内层 `if` 解决次要矛盾，从而将复杂问题分解。
3. **提升代码可读性**：良好的嵌套结构能让代码逻辑与人类的自然思考过程保持一致，是编写高质量、可维护代码的重要技巧。
4. **注意代码缩进**：在编写嵌套代码时，必须使用规范的缩进，以清晰地展示代码的层级关系，否则代码将变得难以阅读和维护。