# C++ 讲义：多分支选择（`if-else if-else`）

## 一、 处理“多选一”与“互斥”

### 1.1 从二元选择到多元选择

我们已经掌握了：

- `if` **结构（单分支）**：解决“做或不做”的问题。
- `if-else` **结构（双分支）**：解决“二选一”的问题。

然而，程序设计中的决策往往涉及三个或更多的选项。`if-else if-else` 结构正是为了解决这类**多选一**的场景而设计的。理解该结构的关键，在于深刻理解**“互斥”**这一概念。

### 1.2 “互斥”：`if-else if-else` 的适用点

- **互斥场景（适用 `if-else if-else`）：交通信号灯**
  交通信号灯的状态是典型的互斥关系：在任何特定时刻，交通灯只能处于其中一种状态(红、黄、绿)，绝不可能同时既是红灯又是绿灯。**当面临必须“多选一”的排他性选项时，应使用多分支结构。**
- **非互斥场景（不适用多分支，应使用多个独立 `if`）：自助餐取餐**
  假设在自助餐厅，有烤鸡翅、牛排、三文鱼、沙拉四个选项。这些选项是不互斥的：你拿了烤鸡翅，并不会阻止你继续拿牛排和沙拉。这些条件可以同时成立。在这种“多选多”的场景下，必须使用多个独立的 `if` 语句来判断，不能使用 `else if`（因为 `else if` 一旦命中一个，就会跳过后续所有选项）。

## 二、 语法结构与执行流程

### 2.1 语法结构与规则

```
if (条件1) {
    // 仅当 条件1 成立时执行
} else if (条件2) {
    // 当 条件1 不成立，且 条件2 成立时执行
} else if (条件n) {
    // ... 可以根据需要添加任意多个 else if 分支 ...
} else {
    // 当以上所有条件都不成立时，执行此保底操作（此分支可省略）
}
```

**结构非常灵活：**

- `else if` **可以有无限个**：根据实际决策的需要，你可以在 `if` 之后添加任意多个 `else if` 分支，以覆盖所有可能的互斥选项。
- `else` **是可以省略的**：最后的 `else` 分支并非强制要求。如果省略了 `else`，且上方所有的 `if` 和 `else if` 条件评估均为 `false`，程序将直接跳过整个多分支结构，不执行其中的任何代码，继续向下运行。

### 2.2 执行顺序

1. **自上而下，顺序判断**：程序会严格按照从上到下的顺序，逐一评估每个 `if` 或 `else if` 的条件。
2. **首次命中，立即跳出**：一旦某个条件被评估为 `true`，程序将执行其对应的代码块。执行完毕后，**立即跳出整个多分支结构**，后续所有的 `else if` 和 `else` 分支都会被忽略。

3. `else` **隐含的条件**：每一个 `else if` 语句都隐含了一个前提——**它前面的所有 `if` 和 `else if` 条件均为 `false`**。这是编写简洁代码的关键。

## 三、 实战：BMI 肥胖问题评估 (书本P40 例3-7)

### 3.1 任务目标

根据输入的体重 `m` (kg) 和身高 `h` (m)，计算身体质量指数 `BMI = m / (h * h)`，并根据以下规则输出结果：

- **规则1**：BMI < 18.5，输出 "Underweight"。
- **规则2**：18.5 <= BMI < 24，输出 "Normal"。
- **规则3**：BMI >= 24，输出 BMI 数值并换行输出 "Overweight"。

### 3.2 代码实现与逻辑分析

```
int main() {
    double m, h, bmi;
    cin >> m >> h;
    bmi = m / (h * h);
    if (bmi < 18.5) {
        cout << "Underweight";
    } else if (bmi < 24) {  // 利用隐含的条件简写为 bmi < 24
        cout << "Normal";
    } else {                // 兜底选项
        cout << bmi << endl;
        cout << "Overweight";
    }
    return 0;
}
```

### 3.3 代码解读

在判断"正常"范围时，我们**没有写** `else if (bmi >= 18.5 && bmi < 24)`，而是简写为 `else if (bmi < 24)`。

**原因**：正是利用了多分支结构的**"隐含条件"**特性。程序能执行到这个 `else if`，必然是因为第一个 `if (bmi < 18.5)` 判断为假。因此，被筛选到第二层的 BMI 数据，已经天然满足了 `bmi >= 18.5` 这个条件，无需再次判断，使得代码更简洁。

## 四、 知识点总结

1. **明确结构适用性**：
   处理多个且选项之间**完全互斥**的决策时，使用 `if-else if-else`。处理选项可共存（如自助餐）的决策时，必须使用多个独立的 `if`。
2. **结构灵活**：
   一个多分支结构由 1个 `if` 起头，中间可以有 **0个或任意多个** `else if`，结尾可以有 **0个或1个** `else` 组成。虽然语法上 `else` 可以省略，但建议保留作为"保底"机制，处理其他所有未被匹配到的情况。
3. **重视书写顺序**：
   在多分支结构中，通常应优先将**最严格、最特殊**的条件放在前面，并善于利用结构自带的隐含的条件来精简后续的判断逻辑。