C++ `if` 条件分支 讲义

1. 引言

- **顺序结构的局限**
  - 在之前的课程中，我们学习了 C++ 的：变量（我们的数据容器）、运算符（进行计算和比较的工具）以及输入输出。我们写的代码，就像一条的单行道，从 `main` 函数的第一行开始，按照顺序，一句话一句话地执行下去，直到终点。

但是，现实世界要复杂许多。我们每天都要做无数个决定：

- **如果**我饿了，**就**去吃饭。
- **如果**考试分数大于等于90分，**就**是优秀；**否则如果**大于等于80分，**就是**良好...
- 我们之前所写的代码都遵循一种**顺序结构**。就像下图的流程一样，程序从起点出发，一条路走到黑，中间没有任何改变方向的机会。

  `int k = 1;` -> `k += 10;` -> `k++;` -> `++k;` -> `k %= 10;` -> `k += 6;`

  无论运行多少次，只要初始值不变，结果永远是固定的。这种结构对于简单的计算是足够的，但它无法应对任何需要随机应变的场景。

  如果想让我们的程序变得“智能”，能够模拟现实世界的逻辑，能够根据不同的情况做出不同的反应，那么我们就必须教会它如何**做选择**。

- **问题引入：让程序学会思考**

  要让程序变得有用，它必须能够响应变化。比如：

  - **网站登录**：**如果**用户输入的密码**等于**正确的密码，**就**跳转到主页；**否则**，就提示“密码错误”。
  - **游戏逻辑**：**如果**玩家的生命值**小于或等于**0，**就**显示“游戏结束”。
  - **自动售货机**：**如果**投入的金额**大于或等于**商品价格，**就**出货并找零。

  这些场景的核心，都在于一个词——**如果 (if)**。

---

2. 核心思想 (什么是 `if`？)

- **思路铺垫**

  如果说顺序结构是一条单行道，那么**分支结构就像是在程序中设置了一个个十字路口。**当程序执行到这个路口时，它会根据一个特定的条件，来决定接下来该怎么走。

  而 `if` 语句，就是我们在 C++ 中设置这个十字路口的工具。

- **`if` 语句的形态**

  形态一：`if` (单选 - 如果...就...)

  这是最简单的一个形态，它只关心**“如果条件满足，该做什么。**

  - **语法结构**：

    ```
    if (条件表达式) {
        // 如果 '条件表达式' 的结果为 true，
        // 就执行这里的代码。
    }
    // 如果为 false，则直接跳过花括号内的所有代码。
    ```

- `条件表达式`：这里的条件，通常是一个**关系运算**（如 `age > 18`）或**逻辑运算**（如 `score >= 60 && is_ok == false`）。它的最终计算结果必须是一个**布尔值**：`true`（真）或 `false`（假）。
- `{ ... }`：花括号内的部分，就是满足条件时要执行的代码。

形态二：`if-else`（二选一 - 如果...就...否则...）

很多时候，我们不仅关心条件满足时做什么，也关心条件**不满足**时做什么。

- **语法结构**：

```
if (条件表达式) {
    // 条件为 true 时执行这里。
} else {
    // 条件为 false 时执行这里。
}
```

- **互斥关系**：`if` 代码块和 `else` 代码块是"有你没我"的。程序在一次判断中，**有且仅有**一个代码块会被执行。

---

3. 实例

案例：判断奇偶数（演示 `if-else`）

- **问题**：输入一个整数，判断它是奇数还是偶数。
- **逻辑**：一个数如果是偶数，那么它被2整除的余数必然是0。
- **代码实现**：

```
if (number % 2 == 0) { // 条件：number除以2的余数是否为0？
    cout << "这是一个偶数。" << endl;
} else {
    cout << "这是一个奇数。" << endl;
}
```

---

4. 总结

1. `if` 语句赋予了程序**决策能力**，是构建复杂逻辑的基础。
2. `if` 用于单向判断，`if-else` 用于二选一，`if-else if-else` 用于多选一。
3. `if` 括号中的条件必须是一个最终能得出 `true` 或 `false` 的表达式。
4. `if - else if` 是**从上到下、依次判断、择一执行**的。
5. 注意区分一个等号(**赋值运算符**)和两个等号(**关系运算符**)的区别！**判断是否相等，一定要用两个等号 `==`！**
    - 如果你写了这样的条件判断语句：`if (score = 100)`，那么会先执行 `= 100` 的赋值语句，让左侧 `score` 变量的值变为 `100`。最终括号里还剩下 `if(score)`，也就等价于 `if(100)`，而在 C++ 中，任何非零的整数在 `if` 的条件中都会被当作 `true`。所以，`if(100)` 就等同于 `if(true)`，这个 `if` 语句块**永远**都会被执行！