C++ 逻辑表达式的构建 讲义

一、 目标

我们的目标，是将生活中的复杂规则（自然语言）准确地翻译成计算机能懂的、没有歧义的逻辑表达式（代码）。

二、 翻译的步骤公式

无论规则多复杂，都可以遵循这三步：

1. **拆解**：将一句长长的规则，拆分成一个个最小的、不可再分的**独立判断条件。**
    - 例：“一个数字 `x` 是大于4的偶数” -> 拆成 `x > 4` 和 `x是偶数`。
2. **观察连接词**：找出连接这些独立条件的**逻辑关系词。**
    - “并且”、“同时”、“但” -> 对应 C++ 的 `&&`。**全真才为真，一假即为假。**
    - “或者”、“至少一个” -> 对应 C++ 的 `||`。**一真即为真，全假才为假。**
    - “不”、“不是” -> 对应 C++ 的 `!`，将一个条件的结果**取反。**
3. **组合**：用 C++ 的逻辑运算符将这些条件按顺序组合起来。

三、 优先级

计算机在计算表达式时，并不是简单地从左到右。也就是存在**运算符优先级。记忆口诀：非算关与或**

- **基本规则**：`!` (逻辑非) 的优先级最高，其次是 `&&` (逻辑与)，最后是 `||` (逻辑或)。
- **使用括号 `()` 明确意图**
    - 为了避免优先级带来的混淆和错误，当一个表达式中同时包含多个逻辑运算符的时候时，**使用括号 `()` 来明确你的逻辑分组**，这既能保证程序正确，也让代码更容易阅读。
    - 例如：如果你数学**或者**英语考了100分，**并且**作业都写完了，我们就去游乐园。
    - 错误写法: `数学100分 || 英语100分 && 作业都写完了` （错误原因： `&&` 的优先级比 `||` 高）
    - 正确写法: `（数学100分 || 英语100分） && 作业都写完了`。

四、 布尔变量

以后当我们写条件表达式的时候，如果一个判断条件的结果，已经被保存在一个布尔（bool）变量里了，我们就可以直接根据它的语义来去使用。

假设我们已经将两个判断条件，分别存入了布尔变量 `a` 和 `b` 中：

- `bool a = (考试成绩 > 80);` // a 代表“考试成绩大于80分”这件事
- `bool b = (综合成绩 > 90);` // b 代表“综合成绩大于90分”这件事

当我需要写一个条件表达式来判断：“考试成绩大于80分 **并且** 综合成绩大于90分”时，我们就可以像翻译句子一样，把自然语言和我们的代码进行一一对照：

- “考试成绩大于80分” → 这正是我们的布尔变量 `a` 所代表的含义。
- “并且” → 在代码中对应的就是逻辑运算符 `&&`。
- “综合成绩大于90分” → 这正是我们的布尔变量 `b` 所代表的含义。

然后，把它们按顺序串联起来，就得到了最终的表达式：`a && b`。

这种方式让我们的代码逻辑和自然语言逻辑完全对应。而不需要通过 `a == true && b == true` 重复进行判断。

五、 整除的逻辑表达式

在编程中，我们经常需要判断一个数能否被另一个数整除。这在判断奇偶性、倍数关系时非常有用。

- **工具**：求余运算符 `%`。`a % b` 的意思是计算 `a` 除以 `b` 之后的**余数**。
- **逻辑**：如果 `a` 能被 `b` 整除，那么 `a` 除以 `b` 之后**没有余数**，所以 `a % b` **的结果就是** `0`。
- **实例**：
  - `9 % 3` 的结果是 `0`，所以 `9 % 3 == 0` 为 `true`，代表9是3的倍数。
  - `10 % 3` 的结果是 `1`，所以 `10 % 3 == 0` 为 `false`，代表10不能被3整除。

六、 构建实例

实例1：闰年判断

- **规则**：年份能被4整除**但**不能被100整除，**或者**能被400整除。
- **第一步：拆解**
  - 条件A: 年份能被4整除 -> `year % 4 == 0`
  - 条件B: 年份不能被100整除 -> `year % 100 != 0`
  - 条件C: 年份能被400整除 -> `year % 400 == 0`
- **第二步：观察连接词**
  - 条件A和B之间是“但”（也就是“并且”）的关系。
  - (A和B的整体) 与 条件C之间是“或者”的关系。
- **第三步：组合（使用括号明确优先级）**

```
bool a = ((year % 4 == 0) && (year % 100 != 0)) || (year % 400 == 0);
```

实例2：数字性质筛选

- **规则**：一个数字 `x` 要满足“是3的倍数，但不是偶数”。
- **第一步：拆解，并用布尔变量存起来**
  - 条件A：“是3的倍数”

```
bool b = (x % 3 == 0); // "是3的倍数" 的结果存在布尔变量 b 里
```

  - 条件B：“是偶数”

```
bool c = (x % 2 == 0); // "是偶数" 的结果存在布尔变量 c 里
```

- **第二步：观察连接词**
  - 条件A 和 “不是条件B” 之间是“但”（“并且”）的关系。
- **第三步：组合（使用布尔变量）**

```
// 条件A和条件B的结果已经储存在布尔变量中了，可直接根据语义去使用。
bool d = b && !c;
```