第三章 关系、逻辑与分支结构

本章内容

- 3.1 关系表达式与逻辑表达式
- 3.2 分支语句
- 3.3 分支嵌套
- 3.4 分支程序设计案例

3.1 关系表达式与逻辑表达式

核心知识

关系运算用于比较两个值，结果是 `bool` 类型，即 `true` 或 `false`。而逻辑运算符可以把多个真假条件组合成更复杂的判断。

C++ 在需要判断真假的位置，会把数值 `0` 视为假，把非零值视为真。`cout` 默认把 `true` 输出为 `1`，把 `false` 输出为 `0`。

关系运算符

运算符	含义	示例
`==`	等于	`score == 100`
`!=`	不等于	`x != 0`
`>`	大于	`a > b`
`<`	小于	`a < b`
`>=`	大于等于	`age >= 18`
`<=`	小于等于	`score <= 60`

关系表达式示例：

```
bool result1 = (10 > 5);   // true
bool result2 = (3 == 4);   // false
bool result3 = (7 != 0);   // true
```

逻辑运算符

表达式	含义	何时为真
`A && B`	A 并且 B	A、B 都为真
`A || B`	A 或者 B	A、B 至少一个为真
`!A`	不是 A	A 为假

自然语言与 C++ 表达式对照：

```
// x 在闭区间 [10, 20] 内
(x >= 10) && (x <= 20)

// x 小于 0 或大于 100
(x < 0) || (x > 100)

// x 是 3 的倍数，但不是偶数
(x % 3 == 0) && (x % 2 != 0)

// x 不等于 10
!(x == 10) // 或者 x != 10
```

构建逻辑表达式时，可以按三个步骤完成：

1. **拆解条件**：把长规则拆成独立的小判断。
2. **寻找连接词**：“并且”对应 `&&`，“或者”对应 `||`，“不”对应 `!`。
3. **组合表达式**：使用括号明确每一组条件之间的关系。

运算优先级

常用优先级从高到低可概括为(**非算关与或**)：

```
括号 → 逻辑非 ! → 算术运算 → 关系运算 → && → ||
```

对于一些复杂表达式应主动使用括号，不要只依赖优先级记忆，例如：

```
(math == 100 || english == 100) && homeworkFinished
```

短路求值

对于 `A && B`：

- 如果 A 为假，整个表达式一定为假，B 不再执行。

对于 `A || B`：

- 如果 A 为真，整个表达式一定为真，B 不再执行。

```
int x = 0;
int y = 10;

bool a = (x != 0) && (100 / x > 2);
// 左侧为 false，右侧不执行，避免除以 0

bool b = (y > 0) || (++y > 20);
// 左侧为 true，右侧不执行，y 仍为 10
```

短路既能减少不必要的计算，也能保护右侧表达式不在非法条件下执行。

典型例题：闰年表达式

闰年规则为：年份能被 4 整除但不能被 100 整除，或者能被 400 整除。

拆解：

- A：`year % 4 == 0`
- B：`year % 100 != 0`
- C：`year % 400 == 0`

组合：

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

常见错误

- 判断相等时写成 `=`。`=` 是赋值，`==` 才是比较。
- 写成数学连写 `5 < x < 10`。正确写法是 `x > 5 && x < 10`。
- 把逻辑与、逻辑或写成 `&`、`|`。应使用 `&&`、`||`。
- 区间题漏掉端点的等号。
- 复杂条件没有括号，导致实际的计算优先级与题意不同。
- 在逻辑表达式右侧使用 `x++` 等有副作用的操作，却忽略短路可能导致它不执行。

3.2 分支语句

核心知识

分支结构让程序根据条件选择不同的执行路径：

- `if`：只关心条件成立时需要做什么。
- `if-else`：处理二选一。
- `if-else if-else`：处理多个互斥选项，只执行首次满足的分支。
- 多个独立 `if`：每个条件分别判断，可能执行多个分支。
- `switch`：处理同一个整型或字符型表达式的多路等值选择。

单分支和双分支

```
if (condition) {
    // condition 为 true 时执行
}
```

```
if (condition) {
    // 条件成立时执行
} else {
    // 条件不成立时执行
}
```

奇偶判断：

```cpp
if (number % 2 == 0) {
    cout << "偶数";
} else {
    cout << "奇数";
}
```

多分支

```cpp
if (condition1) {
    // 第一种情况
} else if (condition2) {
    // 第一种情况不成立，第二种情况成立
} else {
    // 以上条件都不成立
}
```

多分支按照从上到下的顺序判断。一旦某个条件为真，程序执行对应代码块，然后跳过整条分支链。因此，条件书写顺序会影响结果。

例如对于成绩分级的实例，应首先判断更高的分数：

```cpp
if (score >= 90) {
    cout << "A";
} else if (score >= 80) {
    cout << "B";
} else if (score >= 60) {
    cout << "C";
} else {
    cout << "D";
}
```

而且对于多分支而言，后面的条件隐含了前面条件不成立。例如在执行 `score >= 80` 的分支时，已经知道 `score < 90`。

独立 `if` 与 `else if`

多个独立 `if` 属于“多选多”：

```cpp
if (x % 2 == 0) {
    cout << "even ";
}
if (x % 3 == 0) {
    cout << "multiple of 3";
}
```

当 `x` 为 6 时，两个条件都成立，两段代码都会执行。

`if-else if` 属于“多选一”：前面的分支一旦命中，后面的分支不再判断。

`switch` 多路选择

`switch` 适合对同一个表达式进行多个等值匹配：

```
switch (expression) {
    case constant1:
        // expression 等于 constant1
        break;
    case constant2:
        // expression 等于 constant2
        break;
    default:
        // 没有任何 case 匹配
        break;
}
```

规则：

- `switch` 后的 `expression` 表达式应为整型、字符型的值。
- `case` 后的 `constant` 所匹配的值必须是常量，不能写范围条件，不能写变量。
- `break` 会立即跳出 `switch`。
- `default` 是可选的保底分支，作用类似多分支最后的 `else`。
- 缺少 `break` 时，程序会从匹配位置继续向下执行直到遇到 `break` 为止，这种情况称为** `case` 穿透 **。

在一些特殊情况，多个 `case` 可以有意使用穿透来共用一段逻辑：

```
switch (letter) {
    case 'a':
    case 'e':
    case 'i':
    case 'o':
    case 'u':
        cout << "元音";
        break;
    default:
        cout << "非元音";
        break;
}
```

常见错误

- 在 `if` 条件后误写分号：

  ```
  if (x > 0); { // 错误：分号提前结束了 if
      cout << x;
  }
  ```

- 省略花括号后添加第二条语句，误以为两条语句都受 `if` 控制。
- 把多选一写成多个独立 `if`，造成多个分支同时输出。

- 把允许多个结果的情况写成 `else if`，导致后续条件被跳过。
- 多分支先写宽松条件，导致严格条件永远没有机会命中。
- 用 `switch` 处理 `score >= 60` 等范围条件。
- `case` 后写变量、范围或重复的常量值。
- 漏写 `break`，造成意外穿透。

3.3 分支嵌套

核心知识

分支嵌套，是在一个 `if`、`else if` 或 `else` 的代码块内部再放入一个完整的分支结构。

嵌套适合“先判断大前提，再处理细分情况”的问题：

- 外层分支确定主要类别或进入资格。
- 内层分支只在外层条件成立后继续判断。
- 如果外层条件不成立，内层条件不会执行。

语法与执行流程

```
if (外层条件) {
    if (内层条件) {
        // 外层条件和内层条件都成立
    } else {
        // 外层条件成立，但内层条件不成立
    }
} else {
    // 外层条件不成立
}
```

执行过程：

1. 先判断外层条件。
2. 外层条件为假时，跳过整个外层代码块。
3. 外层条件为真时，进入代码块，再判断内层条件。
4. 内层分支执行完成后，程序继续执行嵌套结构后面的代码。

典型例题：会员折扣

商场规则如下：

- 非会员按原价结算。
- 会员消费满 500 元打 8 折。
- 会员消费不足 500 元打 9 折。

```
#include <iostream>
using namespace std;

int main() {
    double amount;
```

```
    bool isMember;
    cin >> amount >> isMember;

    double payment = amount;

    if (isMember) {
        if (amount >= 500) {
            payment = amount * 0.8;
        } else {
            payment = amount * 0.9;
        }
    }

    cout << payment;
    return 0;
}
```

说明： 外层先判断会员资格，只有会员才进入内层判断消费金额。非会员不进入内层，`payment` 保持原价。

常见错误

- 缩进混乱，无法看清 `else` 与哪个 `if` 配对。
- 外层条件不成立时仍然期待内层判断执行。
- 把所有条件都塞进多层嵌套，使代码难以阅读。能用清晰逻辑表达式解决时，不必强行嵌套。
- 忘记为各层代码块补齐花括号。
- 在不同层级重复或遗漏条件，导致某些情况没有处理。

3.4 分支程序设计案例

解决分支题时，可以采用四步流程：

1. **找到分岔点**：哪些条件会改变程序执行路径？
2. **写出条件表达式**：把自然语言准确翻译为关系或逻辑表达式。
3. **选择分支结构**：单选、二选一、多选一、多选多、嵌套或 `switch`。
4. **实现各分支代码**：在每个分支写入对应处理代码。

案例一：BMI 分类

输入体重 `mass`（kg）和身高 `height`（m），计算 BMI。小于 18.5 输出 `Underweight`；小于 24 输出 `Normal`；否则输出 `Overweight`。

```
#include <iostream>
using namespace std;

int main() {
    double mass, height;
    cin >> mass >> height;

    double bmi = mass / (height * height);

    if (bmi < 18.5) {
```

```
        cout << "Underweight";
    } else if (bmi < 24) {
        cout << "Normal";
    } else {
        cout << "Overweight";
    }

    return 0;
}
```

第二个分支隐含了第一个条件不成立，即 `bmi >= 18.5`，所以只需继续判断 `bmi < 24`。

案例二：闰年判断

输入年份。如果是闰年，输出该年份；否则输出 `No`。

```
#include <iostream>
using namespace std;

int main() {
    int year;
    cin >> year;

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

    if (leap) {
        cout << year;
    } else {
        cout << "No";
    }

    return 0;
}
```

本题先用关系和逻辑运算构建一个完整条件，再用 `if-else` 根据真假输出。

案例三：简易计算器

输入形如 `7 * 2` 的表达式，根据运算符完成计算。

```
#include <iostream>
using namespace std;

int main() {
    double a, b;
    char op;
    cin >> a >> op >> b;

    switch (op) {
        case '+':
            cout << a + b;
            break;
        case '-':
```

```
            cout << a - b;
            break;
        case '*':
            cout << a * b;
            break;
        case '/':
            if (b == 0) {
                cout << "Error";
            } else {
                cout << a / b;
            }
            break;
        default:
            cout << "Unknown operator";
            break;
    }

    return 0;
}
```

`switch` 负责对字符 `op` 做等值匹配；除法分支中再用嵌套 `if` 检查除数是否为 0。

本章练习

一、表达式练习

用 C++ 表达式表示下列条件：

1. `x` 大于等于 10 且小于等于 20。
2. `x` 小于 0 或大于 100。
3. `n` 是 5 的倍数。
4. `n` 能被 3 整除，但不能被 2 整除。
5. 字符 `ch` 是大写英文字母。
6. 年龄 `age` 不在 7 ~ 15 岁的范围内。
7. 数学或英语至少有一门达到 90 分，并且作业已经完成。变量分别为 `math`、`english`、`homeworkFinished`。
8. `a`、`b`、`c` 三个条件中至少有两个为真。

二、代码阅读

1. 写出程序的输出结果：

```
int x = 8;
if (x > 5) {
    cout << "A";
}
if (x % 2 == 0) {
    cout << "B";
}
```

2. 写出程序的输出结果，并说明原因：

```
int score = 85;
if (score >= 60) {
    cout << "及格";
} else if (score >= 80) {
    cout << "良好";
} else if (score >= 90) {
    cout << "优秀";
}
```

3. 写出 `y` 的最终值：

```
int x = 5;
int y = 10;
bool result = (x > 1) || (++y > 20);
```

4. 写出程序的输出结果：

```
int n = 2;
switch (n) {
    case 1:
        cout << "A";
    case 2:
        cout << "B";
    case 3:
        cout << "C";
        break;
    default:
        cout << "D";
}
```

5. 写出程序的输出结果：

```
bool loggedIn = true;
bool isAdmin = false;

if (loggedIn) {
    if (isAdmin) {
        cout << "管理员";
    } else {
        cout << "普通用户";
    }
} else {
    cout << "未登录";
}
```

6. 判断表达式 `5 < x < 10` 能否正确表示 `x` 在 5 和 10 之间，并说明计算过程。

三、程序改错

1. 改正下面的判断语句：

```
if (score = 100); {
    cout << "Full marks";
}
```

2. 下面的程序想判断 `x` 是否在闭区间 [10, 20] 内，请改正：

```
if (10 <= x <= 20) {
    cout << "Yes";
}
```

3. 下面的成绩分级程序可以正确的区分等级吗?

```
if (score >= 60) {
    cout << "C";
} else if (score >= 80) {
    cout << "B";
} else if (score >= 90) {
    cout << "A";
} else {
    cout << "D";
}
```

四、程序设计

1. **奇偶判断**

   输入一个整数，偶数输出 `Even`，奇数输出 `Odd`。

2. **景点门票**

   某景点全价票 50 元。年龄不超过 6 岁或不低于 60 岁时票价为 25 元，其余情况为 50 元。输入年龄，输出票价。

   - 样例输入：`65`
   - 样例输出：`25`

3. **三个数的最大值**

   输入三个整数，使用分支语句输出其中的最大值。

   - 样例输入：`6 5 2`
   - 样例输出：`6`