程序的控制结构

让程序根据条件走不同的路

HGNU-ACM 实验室算法组

截图配套：9 道例题，19 道练习题

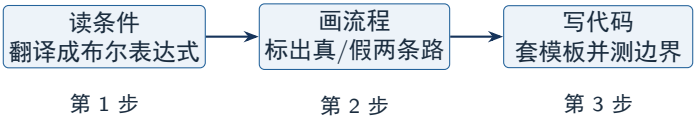




本章目录

先打牢条件与分支，再逐题拆解，最后按索引练习。

本章怎么学：三步走



- 先判断：把中文里的“至少、不超过、同时、或者”变成 `&&`、`||`、`!`。
- 再画图：在纸上写出条件，画出“真走哪边、假走哪边”。
- 后写码：先保证每条语句单独一行，再用边界值手算检查。

本页要点

每节先讲语法，立刻配一个小例子；例题部分按“题意 $\rightarrow$ 推导 $\rightarrow$ 代码”展开。

先试条件，再预测输出

晨练小判断

温度至少 25 度、至多 30 度才适合。把“至少”和“至多”各写成一个判断，用 `&&` 连接。

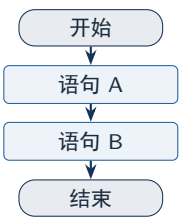
温度	达到 25?	不超过 30?	适合?
24	假	不用判断	否
25	真	真	是
30	真	真	是
31	真	假	否

自查：若把“至少”错写成大于，哪一行会改变？为什么？

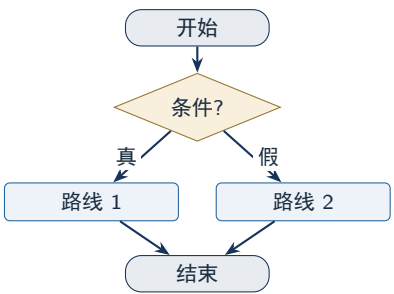
程序为什么需要判断？

- 顺序结构：语句从上到下依次执行，路线固定。
- 分支结构：先算条件，再决定走哪一段。
- 本章讲选择结构；下一章讲循环结构。

顺序结构



分支结构



bool：真和假也是一种数据

- C++ 用 bool 类型保存真假，只有 true 和 false 两个字面量。
- 输出时默认显示 1/0；加上 boolalpha 显示 true/false。
- 比较表达式的结果就是 bool，可以直接放进 if。

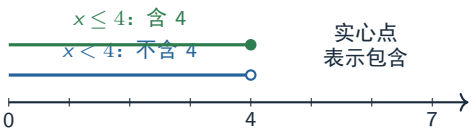
```
bool ok = true;
bool bad = false;
cout << ok << ' ' << bad << '\n';
cout << boolalpha << ok << '\n';
if (ok) {
    cout << "go" << '\n';
}
```

本页要点

0 为假，非 0 为真；初学阶段坚持写 true/false 和比较表达式，不依赖数字。

关系运算符：比较两个值

运算符	含义	例子
<	小于	3 < 5 为真
<=	小于等于	5 <= 5 为真
>	大于	3 > 5 为假
>=	大于等于	5 >= 6 为假
==	等于	5 == 5 为真
!=	不等于	5 != 5 为假



本页要点

边界点用空心圈表示“不含”，实心圈表示“含”；写条件时先看端点归哪一边。

关系表达式的值与优先级

- 关系运算的结果是 `bool`：`5 > 3` 为真，`5 == 3` 为假。
- 优先级从高到低：`!`、算术、关系、`&&`、`||`、赋值。
- 例：`a + 1 < b * 2` 先算加法和乘法，再比较。
- 不确定时加括号，括号最稳。



本页要点

括号可以改变结合顺序；不确定时给每个比较加括号。

关系表达式的小例子

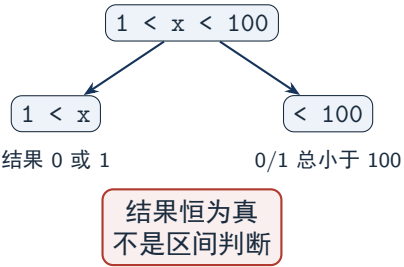
```cpp
int a = 7, b = 3;
cout << (a > b) << '\n'; // 1
cout << (a == b) << '\n'; // 0
cout << (a + 1 < b * 2) << '\n'; // 0
```

本页要点

关系表达式可以直接输出；输出的是 `0/1`，不是 `true/false`。

连续比较：数学写法不能照搬

- 数学里的 $1 < x < 100$ 不能直接写成 `1 < x < 100`。
- C++ 先算 `1 < x`，得到 0 或 1，再算（0 或 1）`< 100`，结果永远为真。
- 正确写法：`x > 1 && x < 100`。
- 同理，`a == b == c` 也不是“三个数相等”。



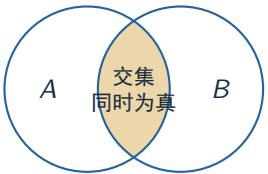
本页要点

每个比较单独写，再用 `&&` 连接；不要写链式不等式。

逻辑与 &&：两个都为真才为真

A	B	A && B
真	真	真
真	假	假
假	真	假
假	假	假

- 两边都是布尔条件，结果才是布尔值。
- 例：`x >= 1 && x <= 100`。



A && B

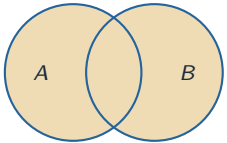
本页要点

“并且”翻译成 &&：缺一个条件都不行。

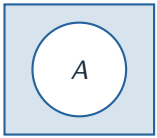
逻辑或 || 与逻辑非 !

A	B	$A \mid\mid B$
真	真	真
真	假	真
假	真	真
假	假	假

A	$!A$
真	假
假	真



并集：任意一边真



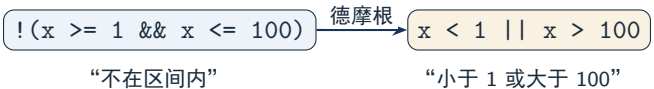
$!A$：圆外区域

本页要点

！把真变假、假变真；它只作用于紧挨着的一个条件。

逻辑运算符的优先级与德摩根律

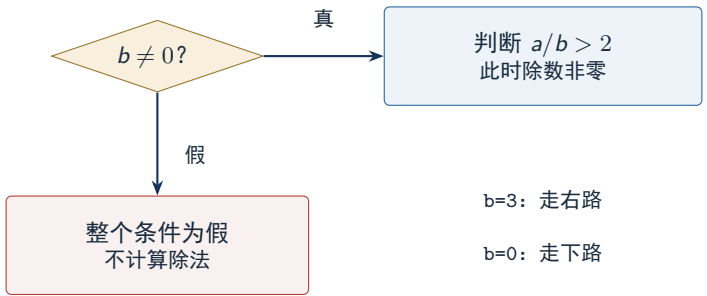
- 优先级：`!` 高于 `&&`，`&&` 高于 `||`。
- `!` 只作用于紧挨着的一个条件：`!a == b` 等价于 `(!a) == b`。
- 德摩根律：`!(a && b)` 等价于 `!a || !b`。
- 德摩根律：`!(a || b)` 等价于 `!a && !b`。



本页要点

取反一个复合条件时，先把每个比较取反，再把 `&&` 和 `||` 互换。

短路求值：&& 的右边可能不执行



本页要点

内置的 && 从左到右判断，左边为假就停止。把除数检查放在除法前。

短路上机：同一份程序试三个输入

```cpp
#include <iostream>

int main() {
    int a = 0, b = 0;
    std::cin >> a >> b;
    bool ok = b != 0 && 1LL * a / b > 2;
    std::cout << (ok ? "yes" : "no") << '\n';
}
```

输入	输出	右边做了什么？
9 0	no	跳过除法
9 3	yes	整除结果为 3
8 3	no	整除结果为 2

运行 `code/short-circuit-demo.cpp`。这里的 `1LL` 让除法按长整数计算。

短路求值：|| 与除零保护

- 计算 A || B 时，若 A 为真，B 不再计算。
- 例：x < 0 || x > 100；左边成立就不必再看右边。
- 除零保护必须把除数检查放在左边：

```
int a, b;
cin >> a >> b;
if (b != 0 && a / b > 2) { // b 为 0 时右边不执行
    cout << "ok" << '\n';
}
if (b == 0 || a / b > 2) { // b 为 0 时右边不执行
    cout << "check" << '\n';
}
```

本页要点

短路只适用于内置的 && 和 ||；不要把关键副作用写在右边。

把中文条件翻译成布尔表达式

- 区间“$1 \le x \le 100$”：`x >= 1 && x <= 100`。
- 闰年：`(y % 4 == 0 && y % 100 != 0) || y % 400 == 0`。
- 三角形：`a + b > c && a + c > b && b + c > a`。
- 偶数/奇数：`x % 2 == 0` / `x % 2 != 0`；后者对负数也成立。

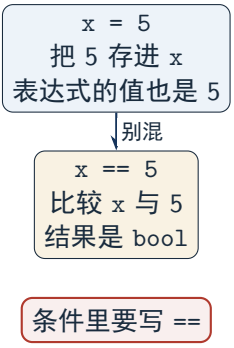


本页要点

先写“并且/或者”，再逐块翻译，最后加括号检查。

= 与 ==：一个等号是赋值

- = 是赋值：把右边的值存进左边，整个表达式的值就是赋进去的值。
- == 是比较：结果是 true 或 false。
- if (x = 0) 先把 0 赋给 x，条件为假；if (x = 5) 条件恒真。
- 防错写法：if (0 == x)；编译时加 -Wall 让编译器提醒。



本页要点

if (x = 0) 不一定报错，但逻辑一定错；看到条件里的单等号就要警惕。

布尔表达式的常见错误

错误写法	问题	正确写法
`1 < x < 100`	链式比较，结果恒真	`x > 1 && x < 100`
`if (x = 0)`	赋值，不是比较	`if (x == 0)`
`if (a & b)`	按位与，不是逻辑与	`if (a && b)`
`if (!x == 0)`	`!` 只作用于 `x`	`if (x != 0)`
`x % 2 == 1`	负奇数会失败	`x % 2 != 0`
`a == b == c`	先算 `a == b`	`a == b && b == c`

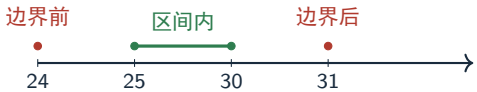
- `&` 和 `|` 是按位运算符；逻辑判断只用 `&&` 和 `||`。
- 不确定优先级时，给每个比较加括号。

本页要点

这六行是本章最常出现的错误，写完代码逐行对照检查。

边界测试：四点法

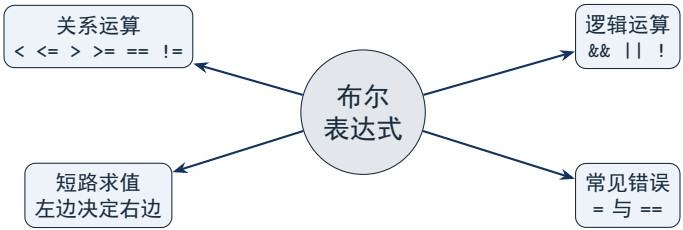
- 对闭区间 $[25, 30]$，测试 $24, 25, 30, 31$。
- 对开区间 $(1, 100)$，测试 $1, 2, 99, 100$。
- 每个边界都测“前一个、边界本身”，再测一个普通值。
- 奇偶、闰年、三角形也要专门测边界和特殊情况。



本页要点

提交前至少手算：最小、最大、边界前、边界本身、普通值。

条件部分小结



- 条件的结果只有 `true`/`false`；组合用 `&&`、`||`、`!`。
- 先算左边，可能短路；写完用边界值测试。

本页要点

下一节：把布尔表达式放进 `if`，让程序真正走出不同的路。

确定多个条件是否互斥

互斥指同一个输入最多满足一个分类。分支链按顺序试条件，命中后跳过后续分支。

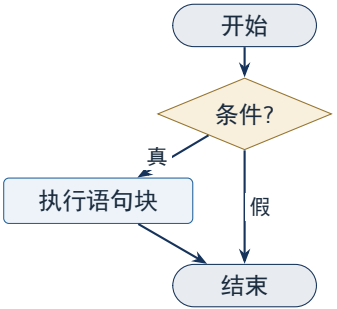
$$\boxed{x = 10} \longrightarrow \boxed{x < 0 \text{ false}} \longrightarrow \boxed{x < 10 \text{ false}} \longrightarrow \boxed{\text{走最后一段}}$$

关键行：先列区间再翻译比较符号，显式检查边界归属；每条路径都给结果赋值。

适用与区别：多个独立 if 可以执行多次，适合多个检验同时触发；分支链最多执行一次。

if 语句：条件为真才执行

- 语法：if（条件）{ 语句块 }。
- 执行流程：先算条件；真则执行花括号内语句，假则整段跳过。
- 无论真假，执行完都继续后面的语句。
- if 只决定“做不做”，不决定“做哪一件”。

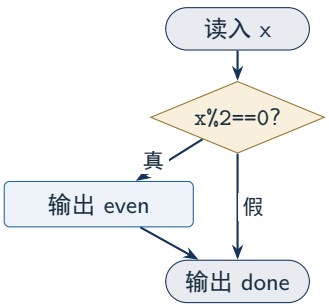


本页要点

条件为假时，if 里面的语句一条也不执行。

小例子：判断偶数 | 流程

- $x = 4$：条件真，输出 even，再输出 done。
- $x = 5$：条件假，跳过 if，只输出 done。
- 条件为假时，if 内的语句一条也不执行。



本页要点

先看条件，再看条件为假时程序走到哪里。

小例子：判断偶数 | 代码

```
int x;
cin >> x;
if (x % 2 == 0) {
    cout << "even" << '\n';
}
cout << "done" << '\n';
```

本页要点

只有一条语句时也可以省略花括号，但本章统一保留花括号，范围更清楚。

花括号与缩进：让分支范围一目了然

- 只有一条语句时可以省略花括号，但初学阶段建议永远写。
- 4 空格缩进；{ 放在条件同一行，} 单独一行。
- 缩进只是给人看的，真正决定范围的是花括号。

```
if (x > 0)
    cout << "positive" << '\n';
cout << "end" << '\n'; // 总是执行
```

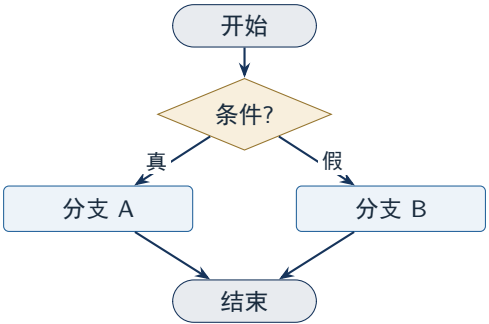
```
if (x > 0) {
    cout << "positive" << '\n';
}
```

本页要点

不确定时加花括号；多写一对括号，比少写造成逻辑错误便宜得多。

if...else：两条路必走一条

- 语法：if（条件）{ ... } else { ... }。
- 条件真走 if 分支，假走 else 分支。
- 两个分支互斥，必定执行且只执行一个。
- 条件互补时用 if...else，不要写两个独立的 if。

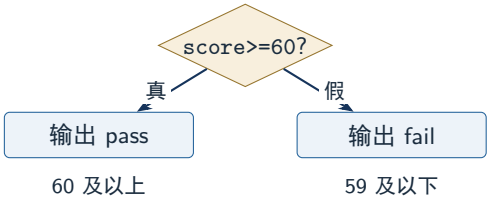


本页要点

else 不需要再写条件，它处理“其余全部情况”。

小例子：及格判定 | 流程

- `score = 60` 走 pass；`score = 59` 走 fail。
- 每个输入恰好得到一行输出。
- 两个分支互斥，不存在“既 pass 又 fail”。



本页要点

条件互补时用 `if...else`，不要写两个独立的 `if`。

小例子：及格判定 | 代码

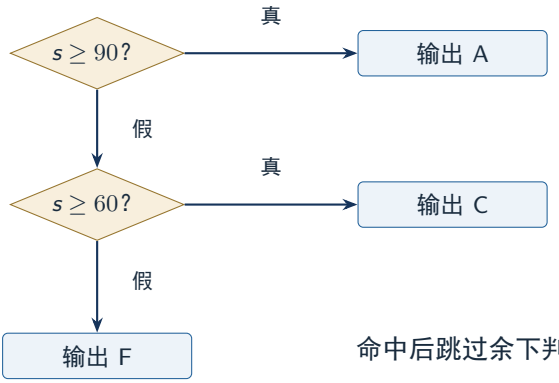
```
int score;
cin >> score;
if (score >= 60) {
    cout << "pass" << '\n';
} else {
    cout << "fail" << '\n';
}
```

本页要点

else 后面不需要写 score < 60，它自动处理其余情况。

`else if`：只进入第一条满足的分支

从上往下问



命中后跳过余下判断

用 $s = 90$ 和 $s = 60$ 各走一次。若把两个判断写成独立的 `if`，90 会满足几条？

小例子：成绩等级

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

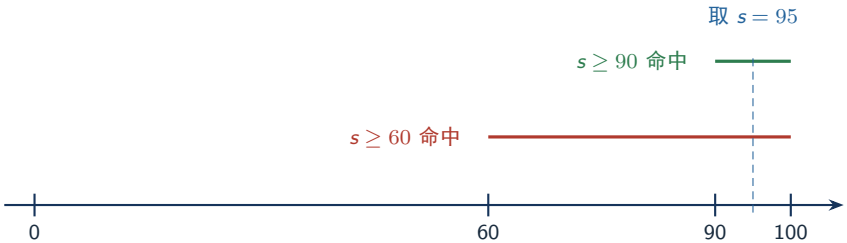
分数	等级
$90 \sim 100$	A
$80 \sim 89$	B
$60 \sim 79$	C
$0 \sim 59$	D

- 条件从高到低写。
- 后面的分支不必再写上限。

本页要点

每个分数只会命中一个分支；顺序写对，条件就可以写得很短。

分支顺序写反会怎样？



先问什么	95 的第一条真条件	输出
先问 $s \geq 60$	第一条，后面的 $s \geq 90$ 没机会	C（错）
先问 $s \geq 90$	第一条，直接得到高分组	A

若一个条件包含另一个，先放范围更小的一条；范围互不重叠时，顺序不改变分类。

分支顺序：错误与正确写法对比

```cpp
if (score >= 60) {
    cout << "C" << '\n';
} else if (score >= 90) {
    cout << "A" << '\n'; // 永远到不了
}
```

错误：95 分先命中 >=60

```cpp
if (score >= 90) {
    cout << "A" << '\n';
} else if (score >= 60) {
    cout << "C" << '\n';
}
```

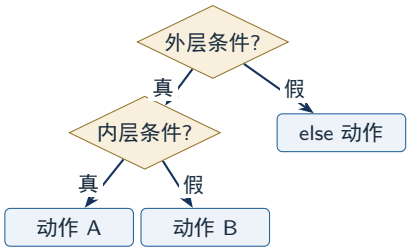
正确：先判断更严格的条件

本页要点

多分支的顺序不是随便排的；先写“更窄”的条件，后写“更宽”的条件。

嵌套 if：条件里面还有条件

- 语法：在一个 `if` 的花括号里再写 `if`。
- 外层条件为真，才进入内层判断；外层为假，内层完全不执行。
- 用途：先分大类，再分小类。
- 嵌套超过三层时，考虑改用 `else if` 或拆分函数。



本页要点

外层是“大门”，内层是“二道门”；大门没开，二道门不会检查。

小例子：三个数的最大值（嵌套版）

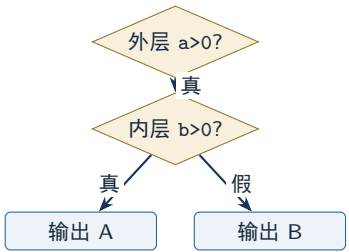
```
int a, b, c, maxv;
cin >> a >> b >> c;
if (a >= b) { // 先比较 a 和 b
    if (a >= c) {
        maxv = a;
    } else {
        maxv = c;
    }
} else {
    if (b >= c) { // 再比较 b 和 c
        maxv = b;
    } else {
        maxv = c;
    }
}
cout << maxv << '\n';
```

悬空 else：else 跟谁配对？

- 规则：else 与最近的、尚未配对的 if 配对，与缩进无关。
- 即使 else 看起来和外层 if 对齐，它仍属于内层。
- 只要不确定，就给每个 if 加花括号。

本页要点

缩进不会改变配对关系；花括号才会。



else 配内层 if

悬空 else：代码示例

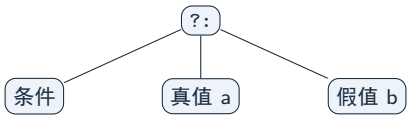
```
int a, b;
cin >> a >> b;
if (a > 0)
    if (b > 0)
        cout << "A" << '\n';
    else
        cout << "B" << '\n'; // 属于内层 if
```

本页要点

这段代码的 else 属于第二层 if；加花括号后意图才没有歧义。

三目运算符：一行写完二选一

- 语法：条件? 真值: 假值。
- 先算条件；真取冒号前的值，假取冒号后的值。
- 它是表达式，可以放在赋值号右边。
- 适合简单二选一；复杂分支仍用 `if`。



本页要点

三目运算符有返回值；`if` 是语句，不能直接写在赋值号右边。

三目运算符：代码示例

```
int a, b;
cin >> a >> b;
int maxv = (a > b) ? a : b;
cout << maxv << '\n';
```

本页要点

先算 a > b；真取 a，假取 b，结果赋给 maxv。

三目运算符的例子与括号

- 绝对值：`int y = (x >= 0) ? x : -x;`。
- 收费单价：`double rate = (w <= 20) ? 1.68 : 1.98;`。
- 输出时必须加括号，否则 `<<` 会先结合：

```
int x;
cin >> x;
int y = (x >= 0) ? x : -x;
cout << (x > 0 ? "positive" : "negative") << '\n';
```

- 不加括号会变成 `(cout << x) > 0 ? ...`，语义完全错位。
- 三目只嵌套一层；超过一层就改用 `if`，可读性更重要。

本页要点

三目运算符适合“二选一取值”，不适合写复杂流程。

分段函数｜核心步骤

模板变化：把一个公式换成三段互斥区间；无论落在哪段都要给结果赋值。

核心步骤：先检查 $x < 0$，否则检查 $x < 10$，其他值进入最后一段。统一在分支之后输出。

完整程序：`code/piecewise.cpp`。

if 家族怎么选

需求	推荐写法
做不做	`if`
二选一	`if...else` 或三目运算符
多选一	`else if` 或 `switch`
大类套小类	嵌套 `if`
按离散值分路	`switch`

- 先画流程图，再选语法；不要一上来就写代码。
- 分支越多，越要把条件和顺序写清楚。

本页要点

`if` 管范围与复杂条件，`switch` 管离散取值；两者可以互相改写。

switch 手推：命中标签以后从哪里停？

设 case 1 打印 one，case 2 打印 two。

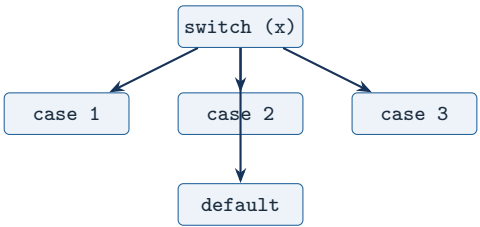
情形	执行路线	屏幕输出
输入 1；两个分支都写 break	进 1，再退出	one
输入 1；1 后漏写 break	进 1，接着执行 2	onetwo
输入 2；1 后漏写 break	直接进 2，再退出	two

本页要点

case 是入口标签。进入后继续执行下面的语句；遇到另一个标签时不会重新比较输入。

switch：按整数/字符的取值分路

- 适用：一个表达式的离散取值，如星期编号、菜单编号、运算符。
- 表达式必须是整型或字符型；case 后必须是常量。
- default 处理所有没列出的值。
- 范围判断（如 $25 \le t \le 30$）更适合 if。

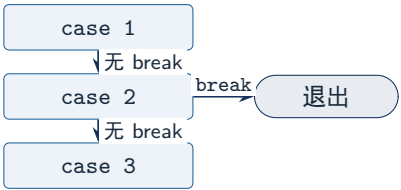


本页要点

switch 是“按值跳转”，不是“按条件判断”。

break 与贯穿：漏写会继续往下走

- 执行流程：计算表达式，跳到匹配的 case。
- 从那里开始一直往下执行，直到遇到 break 或 switch 结束。
- 漏写 break 叫贯穿（fall-through）。
- 多个 case 共享代码时，可以故意不写；初学必须加注释说明。



本页要点

每个 case 结束先写 break，再检查是否故意共享。

break 与贯穿：代码示例

```
switch (x) {
    case 1:
        cout << "one"; // 漏 break
    case 2:
        cout << "two";
        break;
}
```

$x = 1$ 时输出 onetwo：先执行 case 1，再贯穿到 case 2。

本页要点

漏写 break 不会报错，只会安静地多执行后面的分支；测试时要专门覆盖。

多个 case 共享同一段代码

```
int d;
cin >> d;
switch (d) {
    case 1:
    case 2:
    case 3:
    case 4:
    case 5:
        cout << "workday" << '\n';
        break;
    case 6:
    case 7:
        cout << "weekend" << '\n';
        break;
    default:
        cout << "error" << '\n';
}
```

- 连续的 case 标签共用一个语句块。
- 1–5 都跳到 workday。
- 6–7 都跳到 weekend。
- 共享段末尾仍然要 break。

本页要点

共享是合法写法，不是漏写 break；用缩进和注释把意图写清楚。

default：处理所有没列出的值

- default 可选，通常放最后。
- 位置可以任意，但建议放最后，便于阅读。
- 用于输入检查，防止非法值静默通过。
- OJ 题若保证输入合法，default 可以省略；教学代码建议保留。

本页要点

default 是兜底分支，不是“必须执行”的分支。

```
char grade;
cin >> grade;
switch (grade) {
    case 'A':
        cout << "90+" << '\n';
        break;
    case 'B':
        cout << "80+" << '\n';
        break;
    default:
        cout << "unknown" << '\n';
}
```

switch 的限制

项目	限制
表达式类型	整型、字符型、枚举；不能 double、string
case 标签	必须是常量，不能重复、不能是变量
范围写法	标准 C++ 不支持 case 1..5
变量定义	case 内定义带初始化的变量要加花括号

本页要点

需要范围、浮点比较或复杂条件时，回到 if。

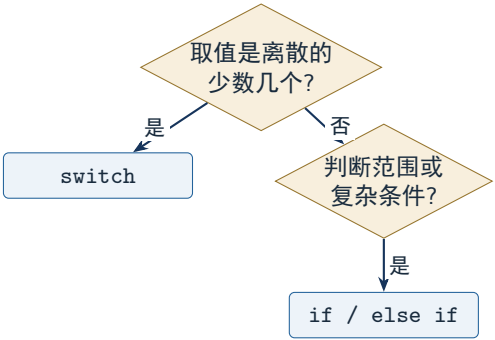
switch 的限制 | case 内定义变量

```
switch (x) {
    case 1: {
        int y = 0; // 花括号限定作用域
        break;
    }
    case 2:
        cout << "two";
        break;
}
```

本页要点

case 内定义带初始化的变量时，用花括号限定作用域。

switch 与 if 怎么选



	switch	if
条件形状	离散值	范围/表达式
可读性	分支多时清晰	分支少时灵活
类型	整型/字符	任意布尔表达式
贯穿	有，需 break	无

本页要点

选择标准：看条件形状，不看个人喜好；能互相改写时选更清楚的那个。

小例子：简单菜单

```
int choice;
cin >> choice;
switch (choice) {
    case 1:
        cout << "add" << '\n';
        break;
    case 2:
        cout << "sub" << '\n';
        break;
    case 3:
        cout << "mul" << '\n';
        break;
    default:
        cout << "bye" << '\n';
}
```

- 输入 1/2/3 进入对应分支。
- 其他值走 `default`。
- 每个分支输出后立即 `break`，不会贯穿。

本页要点

菜单题先列“输入值 → 输出文本”的映射表，再写 `switch`。

switch 常见错误

```
switch (x) {
    case 1:
        int y = 0; // 错误：跨 case 初始化
        break;
    case 2:
        cout << "two"; // 漏 break
}
```

- 漏 break：输出多行。
- case 后写变量：编译错误。
- 重复 case：编译错误。
- 表达式用 double 或 string：编译错误。
- 忘记 default：非法输入无处理。

本页要点

编译错误看行号；逻辑错误（漏 break）靠手算和测试发现。

switch 小结

- 适用离散值；case 后必须是常量。
- break 退出 switch；漏写会贯穿。
- default 兜底；位置任意，建议最后。
- 多个 case 共享代码是特性，不是 bug。
- 每个 case 都可以改写成 if (x == 常量)。



本页要点

先把分支图画对，再写 switch；不确定时用 if 更稳。

例题索引（1/2）

点击题名可跳到讲解页。

- 【例 3.1】偶数
- 【例 3.2】范围判断
- 【例 3.3】三个数
- 【例 3.4】适合晨练
- 【例 3.5】收费
- 【例 3.7】最大的数
- 【例 3.9】星期几

例题索引（2/2）

点击题名可跳到讲解页。

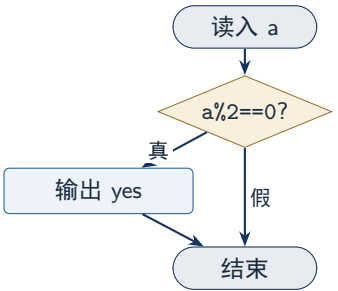
- 【例 3.10】简单计算器
- 【例 3.11】买笔

【例 3.1】偶数 | 题意与推导

题目

输入一个正整数 a。如果它是偶数，输出一行 `yes`；否则不输出任何内容。

1. 偶数的判定条件是 $a \bmod 2 = 0$。
2. 只有条件为真时才执行输出；奇数时程序正常结束，输出为空。
3. 输入保证是正整数，不需要额外判断正负。



样例：$a = 12 \Rightarrow$ yes
$a = 7 \Rightarrow$ 无输出

本页要点

这是“单分支”题：条件为假时什么也不做，不是输出 `no`。

【例 3.1】偶数｜核心步骤

模板变化：使用单分支，条件不满足时不输出。

核心步骤：求对 2 的余数；余数为零才输出 yes。

完整程序：`code/2051.cpp`。

【例 3.2】范围判断｜题意与推导

题目

输入一个整数。若它严格大于 1 且严格小于 100，输出一行 `yes`；否则不输出。

1. 数学中的 $1 < x < 100$ 要拆成两个比较。
2. 两个条件必须同时成立，用 `&&` 连接。
3. 边界 1 与 100 都不满足；测试时连同 2、99 一起检查。



样例：$x = 50 \Rightarrow$ `yes`

$x = 1$ 或 $x = 100$：无输出

本页要点

开区间：两端都不包含，条件写 `x > 1 && x < 100`。

【例 3.2】范围判断 | 核心步骤

模板变化：用逻辑与合并两个条件，必须同时满足。

核心步骤：只在 $1 < x < 100$ 时输出，两个边界都排除。

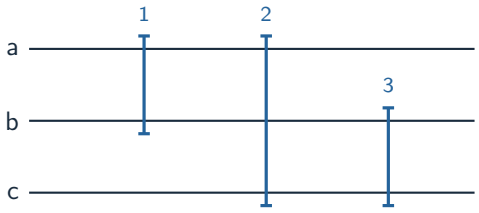
完整程序：`code/2052.cpp`。

【例 3.3】三个数 | 题意与推导

题目

输入三个整数，按从大到小的顺序输出，数之间用一个空格分隔。

1. 目标：让 $a \geq b \geq c$。
2. 依次比较交换：(a, b)、(a, c)、(b, c)。
3. 每一步都把较大的数送到前面，三步后顺序一定正确。



样例：输入 3 1 2
输出 3 2 1

本页要点

这是“排序网络”的最简版本：固定三次比较，不依赖输入顺序。

【例 3.3】三个数｜核心步骤

模板变化：把单条件改为三次独立比较交换；前两次先保证第一个位置最大。

核心步骤：若 $a < b$ 就交换，接着若 $a < c$ 就交换；最后比较并交换 b, c。

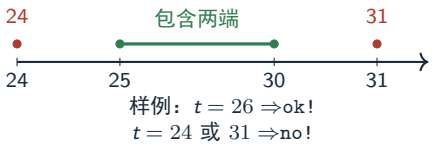
完整程序：`code/2053.cpp`。

【例 3.4】适合晨练｜题意与推导

题目

输入温度 t。若 $25 \le t \le 30$，输出 `ok!`；否则输出 `no!`。

1. 这个区间包含两端，条件是 $t \ge 25$ 且 $t \le 30$。
2. 每个输入都必须得到一个答案，所以使用 `if...else`。
3. 边界 25 和 30 都输出 `ok!`；24 和 31 输出 `no!`。



本页要点

适合/不适合互补，用 `if...else`。

【例 3.4】适合晨练 | 核心步骤

模板变化：使用双分支，无论条件真假都有输出；两个边界均包含。

核心步骤：同时满足 $t \geq 25$ 和 $t \leq 30$ 才适合，其余走另一个分支。

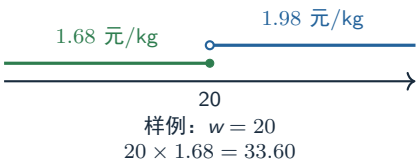
完整程序：`code/2054.cpp`。

【例 3.5】收费 | 题意与推导

题目

行李重量不超过 20 千克时，每千克收费 1.68 元；超过 20 千克时，每千克收费 1.98 元。求总费用，保留两位小数。

1. 以 20 千克为界：$w \le 20$ 用 1.68 元/kg，$w > 20$ 用 1.98 元/kg。
2. $w = 20$ 属于第一段；费用 = 重量 × 单价。
3. 输出用 `fixed` 和 `setprecision(2)`。



本页要点

分段计费题先画数轴，确认边界点归哪一段。

【例 3.5】收费 | 核心步骤

模板变化：分支选择单价，之后统一计算费用。

核心步骤：重量不超过 20 时用 1.68，否则用 1.98。选好单价后乘全部重量。

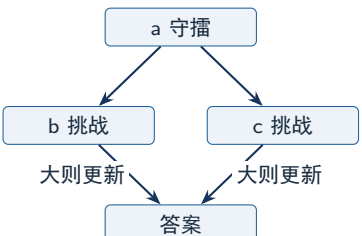
完整程序：`code/2055.cpp`。

【例 3.7】最大的数 | 题意与推导

题目

输入三个实数，输出其中最大的一个。

1. 先令答案为第一个数。
2. 与第二个数比较，再与第三个数比较。
3. 新数严格大于当前答案才更新；例如 $\max(1.01, 2, 3.22) = 3.22$，相等时保留原答案。



本页要点

“打擂台”：先让第一个数守擂，后面每个数挑战一次。

【例 3.7】最大的数｜核心步骤

模板变化：不必把三个数全排序，只维护当前最大。

核心步骤：先把第一个数当答案，后两个数依次与当前答案比较，只在更大时替换。

完整程序：`code/2056.cpp`。

【例 3.9】星期几 | 题意与推导

题目

输入 1 至 7，依次代表 Monday 至 Sunday；其他数字输出 `input error!`。

1. 1–7 各对应一个固定文本，适合 `switch`。
2. 每个 `case` 后写 `break`；无效值进 `default`。
3. 换行统一放在 `switch` 后。

输入	输出
1	Monday
2	Tuesday
⋮	⋮
7	Sunday
其他	input error!

本页要点

标准 `switch` 题：离散编号 $\rightarrow$ 固定文本。

【例 3.9】星期几 | 核心步骤

模板变化：用整数值选择分支，每个合法值只输出一个名称。

核心步骤：按 1 到 7 分别映射到星期名称；每个分支输出后退出，其他值报告错误。

完整程序：code/2057.cpp。

【例 3.10】简单计算器 | 题意与推导

题目

输入两个实数和一个运算符，支持加、减、乘、除。除数为零时输出 `Divided by zero!`，无效运算符输出 `Invalid operator!`。

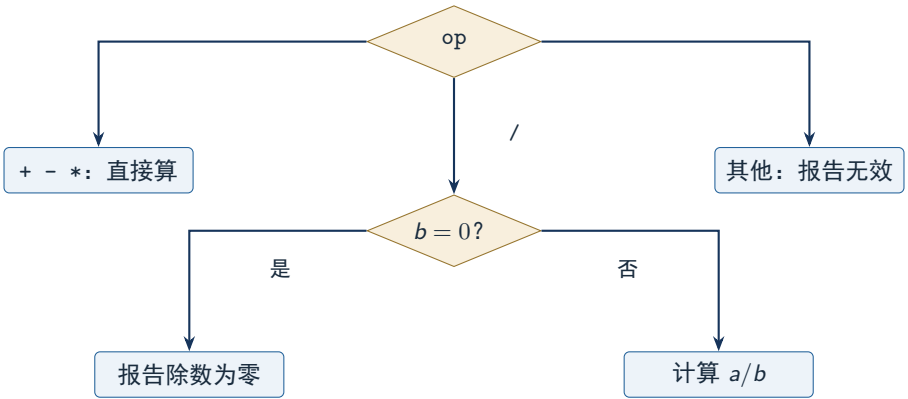
1. 用 `double` 读取 a, b；例如 $5/2 = 2.5$。
2. 除法分支先检查 $b = 0$；其他分支不需要这个检查。
3. `default` 处理非法运算符；换行统一放在 `switch` 后。

运算符	动作
`+`	输出 $a + b$
`-`	输出 $a - b$
`*`	输出 $a \times b$
`/`	$b = 0$ 提示，否则 a/b
其他	`Invalid operator!`

本页要点

异常优先：先判除零和非法运算符。

【例 3.10】简单计算器 | 分支图



用 `80 +` 和 `80 /` 分别走一遍：检查除数只属于除法这条路。

【例 3.10】简单计算器｜核心步骤

模板变化：用字符选择运算，除法分支内部还要判断除数。

核心步骤：加、减、乘直接算；做除法前检查除数是否为零，其他符号报告无效。

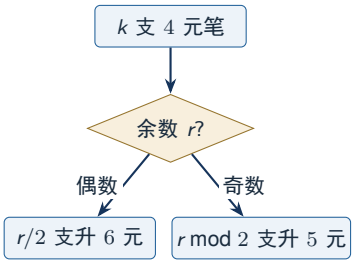
完整程序：`code/2058.cpp`。

【例 3.11】买笔｜题意与推导

题目

有 x 元，钢笔 $6/5/4$ 元。花光且笔数最多；同支数多买 6 元笔。输出三种支数（$x \geq 8$）。

1. 每支至少 4 元，最多买 $k = \lfloor x/4 \rfloor$ 支；余钱 $r = x - 4k \in \{0, 1, 2, 3\}$。
2. $4 \to 6$ 元多花 2 元，$4 \to 5$ 元多花 1 元。
3. 先升 $r/2$ 支到 6 元，再升 $r \bmod 2$ 支到 5 元。



本页要点

样例 $x = 10$：$k = 2, r = 2$，得到 $(1, 0, 1)$。

【例 3.11】买笔｜核心步骤

模板变化：先用整除固定最多支数，再用余钱调整单价构成。

核心步骤：以 4 元为基准，每多用 2 元可升级为 6 元笔，最后剩 1 元可再升级为 5 元笔。

完整程序：`code/2059.cpp`。

例 3.11 | 先保证支数最多，再花光余钱

1. 每支笔至少 4 元，所以支数不可能超过 $k = \lfloor x/4 \rfloor$。
2. 先假设 k 支全是 4 元笔，剩余 $r = x - 4k \in \{0, 1, 2, 3\}$。
3. $4 \to 6$ 元消耗 2 元，$4 \to 5$ 元消耗 1 元。令 $n_6 = \lfloor r/2 \rfloor$、$n_5 = r \bmod 2$。
4. $n_4 = k - n_6 - n_5$。对本题 $x \geq 8$，这三项都非负。

r	n_6	n_5	n_4	总花费
0	0	0	k	$4k$
1	0	1	$k-1$	$4k+1$
2	1	0	$k-1$	$4k+2$
3	1	1	$k-2$	$4k+3$

本页要点

四种余数对应四种升级方案；支数始终是 k，所以一定最多。

例题方法总结

题号	核心条件	易错点
3.1	`a % 2 == 0`	奇数时不要输出
3.2	`x > 1 && x < 100`	两端都不含
3.3	三次比较交换	比较顺序不能乱
3.4	`25 <= t && t <= 30`	两端都包含
3.5	`w <= 20`	20 属于第一段
3.7	打擂台，严格大于才更新	相等时不要更新
3.9	`switch + break`	漏 `break` 会多输出
3.10	`switch` + 除零检查	先判 $b = 0$
3.11	先最多，再升级	余数分配与输出顺序

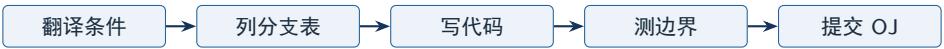
- 拿到新题：先写条件，再画分支，最后测边界。
- 代码排版：每条语句一行，4 空格缩进，关键行写注释。

本页要点

九道例题覆盖了单分支、双分支、多分支、嵌套、三目和 `switch`；后面的练习是它们的组合。

练习安排：先 if，后综合应用

- 第一组 1–8：正负、绝对值、奇偶、两位数、整除，练条件翻译。
- 第二组 9–16：多条件、分段函数、邮资、三角形，练分支顺序与边界。
- 第三组 17–19：闰年、点与正方形、一元二次方程，练综合与特殊情况。



本页要点

每题至少手算三组：样例、边界、特殊情况；不要只交样例。

本章练习题索引 | 1–8 / 19

1. P5039 | 判断数正负（一本通 1039）
2. P5040 | 输出绝对值（一本通 1040）
3. P5041 | 奇偶数判断（一本通 1041）
4. P5042 | 奇偶 ASCII 值判断（一本通 1042）
5. P5043 | 整数大小比较（一本通 1043）
6. P5044 | 判断是否为两位数（一本通 1044）
7. P5045 | 收集瓶盖赢大奖（一本通 1045）
8. P5046 | 判断一个数能否同时被 3 和 5 整除（一本通 1046）

蓝色题名直接进入学校 OJ；本章截图中的练习全部列在索引中。

1. P5047 | 判断能否被 3，5，7 整除（一本通 1047）
2. P5048 | 有一门课不及格的学生（一本通 1048）
3. P5049 | 晶晶赴约会（一本通 1049）
4. P5050 | 骑车与走路（一本通 1050）
5. P5051 | 分段函数（一本通 1051）
6. P5052 | 计算邮资（一本通 1052）
7. P5053 | 最大数输出（一本通 1053）
8. P5054 | 三角形判断（一本通 1054）

蓝色题名直接进入学校 OJ；本章截图中的练习全部列在索引中。

本章练习题索引 | 17–19 / 19

1. P5055 | 判断闰年（一本通 1055）
2. P5056 | 点和正方形的关系（一本通 1056）
3. P5058 | 求一元二次方程（一本通 1058）

蓝色题名直接进入学校 OJ；本章截图中的练习全部列在索引中。

本章检查清单

- 能把中文条件翻译成布尔表达式，并正确处理“并且/或者/不”。
- 能画出 `if`、`if...else`、`else if`、嵌套和 `switch` 的流程图。
- 记得 `=` 与 `==` 的区别，条件里不写单等号。
- `switch` 的每个 `case` 都有 `break`（除非故意共享）。
- 每条语句单独一行，4 空格缩进，关键行有注释。
- 提交前测过最小、最大、边界前、边界本身和普通值。

本页要点

下一章：循环结构——让同一段代码按规律重复执行。