运算符与表达式

判断表达式的类型和值，说明变量值的变化

本讲学习目标

- 判断短表达式的分组、类型和值，区分分组与实际求值先后
- 按任务选择整数商、余数或浮点结果，解释类型转换的位置
- 在采用 ASCII 的示例中，解释字符与编码值，并转换已知的英文字母
- 区分表达式值与对象当前值，说明赋值、自增和自减怎样改变变量值
- 判断比较、逻辑、条件与逗号表达式的结果，区分 = 与 ==
- 对照预期与运行结果，定位短程序的计算偏差并核对局部修正

表达式：从基本运算到操作组合

程序 = 数据对象 + 操作

- 运算符规定对数据执行的运算，表达式将运算符与操作数组合起来。
- 表达式能够描述计算、比较与赋值等操作，扩展程序处理数据的能力。

运算符、操作数与表达式

- 3 + 4 按 C 规则算得 7
- **运算符**：规定某种运算的符号或关键字
  - + 规定加法运算
- **操作数**：某项运算中由运算符处理的表达式
  - 3 和 4 是 + 的两个操作数
- **表达式**：按 C 规则组成、用于计算值或执行操作的代码片段
  - 可以是单个常量 42 或已声明的变量名 sample_count
  - 也可以用运算符组合已有表达式，如 3 + 4

1 表达式的分组、类型和值

从熟悉的计算进入分组、类型和值的判断，再将同一方法用于字符编码。

表达式值与表达式语句

- 从声明 int x = 3; 出发
- **表达式值**：对表达式求值所得到的结果
  - x + 1 求得 4；这个表达式的类型是 int
- **表达式语句**：由表达式和末尾分号组成的语句
  - x + 1; 执行时对 x + 1 求值，但不保存所得的 4
  - 分号 ; 结束这条表达式语句；换行不能代替它
- 表达式算出的值与变量当前保存的值需要分别说明
  - 算出 4 不等于把 4 写回 x；执行后 x 仍为 3
  - int x = 3; 是变量声明与定义，不是语句；有分号不等于就是语句

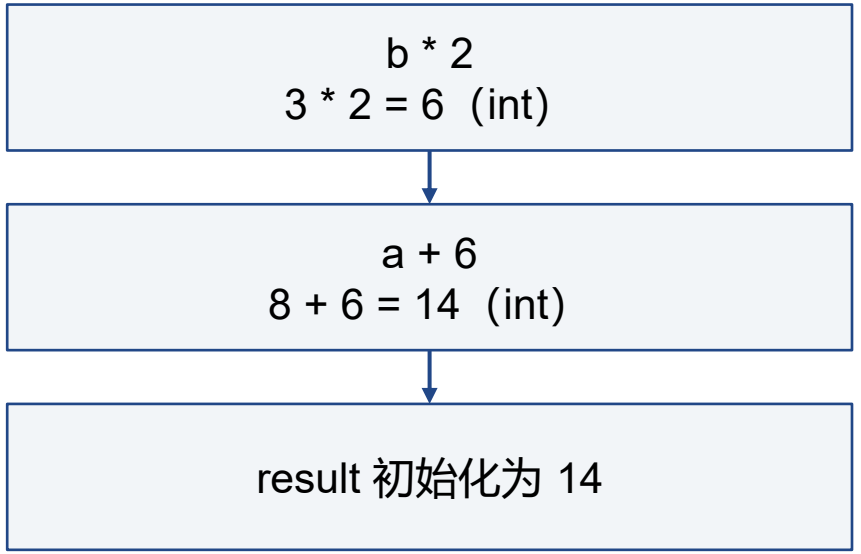
示例：表达式的结构与计算结果

```
1 int a = 8, b = 3;
2 int result = a + b * 2;
```

分组：确定各运算符的操作数，以及子表达式怎样组合

用括号标出结构：a + (b * 2)

乘法子表达式 b * 2 是加法的一个操作数



计算没有修改 a 和 b：之后 a = 8、b = 3

括号、优先级与结合性

- 括号：明确指定哪些部分组成子表达式
  - (2 + 3) * 4 中，(2 + 3) 是乘法的一个操作数
- **优先级**：括号未指定的部分，按运算符优先级确定分组
  - 2 + 3 * 4 按 2 + (3 * 4) 理解
- **结合性**：运算符优先级相同时，按结合方向确定分组
  - 20 - 6 - 3 按 (20 - 6) - 3 理解
- 分组说明子表达式怎样组成，不决定所有操作数实际求值的先后

字符与编码值

- **字符编码**：规定字符如何表示为计算机可存储、处理的数据的规则
- **ASCII**（美国信息交换标准代码）：为英文字母、数字等基本字符分配编码值的标准
- 本讲的示例采用 ASCII
  - 字符 A ↔ 编码值 65；字符 5 ↔ 编码值 53
  - 字符 '5' 与整数 5 不同：53 是字符的编码值
- 'A' ~ 'Z'、'a' ~ 'z'、'0' ~ '9' 三段内部各自连续编码

示例：大写字母转为小写字母

```
1 char upper = 'C';
2 char lower = (char)('a' + (upper - 'A'));
```

本例采用 ASCII，upper 是大写英文字母

大写、小写字母各自连续；同一字母在两段中的位置相同

先求位置差，再移到小写起点

'C' - 'A' 得到位置差 2

'a' + 2 对应小写字符 'c'

字符算术与对象类型

C 字符常量 'C'、'A'、'a' 的类型是 int；本例算术得到整数编码值

显式类型转换：用“(类型名)表达式”明确指定将表达式的值转换为目标类型

(char) 将本例的整数结果转换为 char，用于初始化 lower

upper、lower 都是 char 对象

练习：字符编码与字母转换

```
1 char lower = 'g';
2 char upper = (char)('A' + (lower - 'a'));
```

本例采用 ASCII，lower 是小写英文字母

字符与编码值

'B' 的编码值是多少?

编码值 57 对应哪个字符?

字母位置与转换

lower - 'a' 得到多大的位置差?

upper 的值对应哪个字符?

为什么先减 'a'，再加 'A'?

解析：字符编码与字母转换

```
1 char lower = 'g';
2 char upper = (char)('A' + (lower - 'a'));
```

本例采用 ASCII，lower 是小写英文字母

字符与编码值

'B' 紧接 'A'：65 + 1 = 66

57 = 53 + 4：从字符 '5' 移动 4 位，得到 '9'

字母位置与转换

'g' - 'a' 得到位置差 6

'A' + 6 对应 'G'；upper 保存对应的 char 值

减去原起点、加到目标起点，保留字母在各自序列中的位置

2 整数除法、余数与类型转换

比较除法与转换位置，再用商余数处理数位，最后修正平均值计算。

练习：除法与类型转换

```
7 / 4
7.0 / 4
(double)(7 / 4)
```

- 分别写出三个表达式的类型和值
- 哪些式子会舍去小数？写出判断依据

解析：除法与类型转换

表达式	运算与转换顺序	表达式类型和值
7 / 4	两个 int 相除，舍去小数	int：1
7.0 / 4	7.0 是 double，4 自动转 double	double：1.75
(double)(7 / 4)	先得 int 值 1，再转 double	double：1.0
(double)7 / 4	先把 7 转 double，再除法	double：1.75

- **类型转换**是把一个值转换为另一种类型的值；(double)表达式 显式将表达式值转为 double
- int 与 double 相除时，int 自动转为 double；整数除法后的转换不能恢复已舍去的小数

商和余数

- 把 42 按每组 10 分组：10 + 10 + 10 + 10 + 2 = 42
  - **商**是完整的组数：42 / 10 = 4
  - **余数**是分组后剩下的量：42 % 10 = 2
  - 代入检查：42 = 4 * 10 + 2
- 按 10 分组：n = 10q + d
  - q = n / 10 是商，d = n % 10 是余数，也是个位
  - 非负整数的个位只能是 0 ~ 9
- 本页处理非负整数和正整数除数；% 的操作数须是整数，/ 与 % 的除数都不能为 0

示例：整数的数位拆分与重组

当前 n	个位 n % 10	新 n = n / 10	已读出：低位到高位
507	7	50	7
50	0（原数十位）	5	7、0
5	5	0	7、0、5

- 拆位：n % 10 读个位，n / 10 去个位；中间 0 也保留，新 n 为 0 时本例读完
- 重组（高→低）：乘 10 为新个位留位置，再加下一位，0 也不跳过：5*10+0=50，50*10+7=507

练习：整数的数位拆分与重组

- 逐位拆分 10005，按低位到高位记录
  - 每一步的当前 n、个位 n % 10、新 n = n / 10
  - 中间连续的 0 分别来自哪一步?
  - 当新 n 为 0 时，核对是否已读完所有数位
- 重组给定的百位、十位、个位 3、0、8
  - 写出每次乘 10 再加入下一位的算式与最终整数
  - 解释十位 0 的作用

解析：整数的数位拆分与重组

当前 n	个位 n % 10	新 n = n / 10
10005	5	1000
1000	0	100
100	0	10
10	0	1
1	1	0

- 拆位（低→高）：5、0、0、0、1；重组（高→低）：3*10+0=30，30*10+8=308。0 占十位。

练习：修正平均值计算

```c
int total = 17, count = 5;
double average = total / count;
printf("%.2f\n", average);
```

平均值任务

按当前代码推算的输出：3.00

目标输出：3.40

定位并修正

找出第一次丢失小数的子表达式

只修改必要的代码，说明它怎样改变该次除法的操作数类型

解析：修正平均值计算

```
1 int total = 17, count = 5;
2 double original = total / count;
3 double corrected = (double)total / count;
```

原式 total / count

两个操作数都是 int，除法先得到 int 值 3

存入 double 后为 3.0，按两位小数显示 3.00

修正为 (double)total / count

除法前已有 double 操作数，得到约为 3.4 的 double 值

按两位小数显示 3.40

3 赋值、自增与自减

区分表达式取得的值和语句结束后变量保存的值。

赋值与复合赋值

赋值把右侧表达式的值转换为左侧对象的类型后，存入左侧对象；赋值表达式的值是赋值后该对象的值。

复合赋值把运算与赋值结合起来；如 x += bonus 将 x 与 bonus 相加，再把结果写回 x。

表达式（各式初值：int x=3, bonus=2;）	表达式的值	求值后的 x
x + 1	4	3
x = x + 1	4	4
x = x + bonus	5	5
x += bonus	5	5

求值时，除取得表达式值外还可能影响程序状态，这类影响称为**副作用**，如向对象写入值。表中只有后三式写入 x，bonus 始终为 2。

赋值运算符按右结合分组；例如两个 int 对象 m、n 的 m = n = 2 按 m = (n = 2) 分组，右侧赋值表达式的值供外层赋值使用。

练习：赋值表达式

```
1 /* A */ x = x + bonus;
2 /* B */ x += bonus;
3 /* C */ a = b = 3;
```

分析条件

A、B 分别从 int x = 4, bonus = 3; 开始；C 中 a、b 是两个不同的 int 对象

A、B：普通与复合赋值

记录表达式的值及语句结束后的 x、bonus

指出赋值写入哪个对象

C：连续赋值

用括号标出分组，记录内层与整体的值及语句结束后的 a、b

两层各写入哪个对象？内层表达式值怎样用于外层赋值？

解析：赋值表达式

题目	赋值表达式的值	语句结束后变量的值
A：x = x + bonus;	7	x = 7，bonus = 3
B：x += bonus;	7	x = 7，bonus = 3
C：a = b = 3;	内层与整体均为 3	a = 3，b = 3

A、B 各用原来的 x = 4 与 bonus = 3 相加，将 7 写入 x；赋值表达式也取得 7。

C 按 a = (b = 3) 分组。

内层 b = 3 向 b 写入 3，表达式值为 3。

外层使用内层表达式的值，向 a 写入 3，整体表达式值为 3。

右结合决定分组，不代表一般操作数的实际求值先后。

自增与自减

自增 ++ 将对象的值增加 1；**自减** -- 将对象的值减少 1。

前置形式 ++x、--x 的表达式值是更新后的值；后置形式 x++、x-- 的表达式值是更新前的值。两种形式都会更新 x。

独立语句（各从 int x = 3, y; 开始）	更新表达式取得的值	语句结束后变量的值
y = ++x;	++x 取新值 4	y = 4，x = 4
y = x++;	x++ 取旧值 3	y = 3，x = 4

y 保存更新表达式取得的值，所以两种情况下 y 不同，而最终 x 相同。

练习：自增与自减

```
1 /* A */ y = ++x;
2 /* B */ y = x++;
3 /* C */ y = --x;
4 /* D */ y = x--;
```

分析条件

四种情况分别从 int x = 5, y; 开始，不接着上一题执行

A ~ D：逐项记录

分别写出更新表达式的值，以及语句结束后的 y、x

两组比较

比较 A 与 B、C 与 D：同一对中的最终 x 是否相同？y 是否相同？说明原因

解析：自增与自减

题目（各从 x = 5 开始）	更新表达式取得的值	语句结束后变量的值
A：y = ++x;	新值 6	y = 6，x = 6
B：y = x++;	旧值 5	y = 5，x = 6
C：y = --x;	新值 4	y = 4，x = 4
D：y = x--;	旧值 5	y = 5，x = 4

- A、B 都将 x 增加 1，C、D 都将 x 减少 1；同一对中，前置取更新后的值，后置取更新前的值，因此赋给 y 的值不同。

4 比较、逻辑与组合表达式

从比较与逻辑组合进入条件选值和逗号运算的顺序求值。

比较运算与赋值

比较运算判断两个值的大小或相等关系；关系成立时得到 int 值 1，否则得到 0。

< 小于，<= 小于或等于。

> 大于，>= 大于或等于。

== 相等，!= 不相等。

表达式（分别从 int x = 3, y = 5; 开始）	表达式的值	求值后的 x
x = y	5	5：赋值写入 x
x == y	0	3：比较不写入 x

条件判断中，0 表示假，非 0 表示真；比较的结果只取 0 或 1。

数学中的连续不等式不能直接用作 C 的范围判断。

60 <= score <= 100 按 (60 <= score) <= 100 分组，前半式得到的 0 或 1 再与 100 比较，整体总为真。

逻辑运算与短路

- **逻辑运算**按真假关系组合或改变条件；&&、||、！的结果均为 int 类型的 0 或 1。
  - **逻辑与** &&：两侧都为真，结果才为真。
    - score 在 60～100 内（含端点）：score >= 60 && score <= 100。
  - **逻辑或** ||：至少一侧为真，结果就为真。
    - cmd 是 'a' 或 'b'：cmd == 'a' || cmd == 'b'。
  - **逻辑非** !：将真假取反。
    - finished = 0 时，!finished 为 1；finished 非零时，!finished 为 0。
- **短路求值**：左侧已经决定整体真假时，不再求值右侧。
  - && 左侧为假，整体必为假；|| 左侧为真，整体必为真。
  - score = 50 时，上述范围条件的左侧为假，不再求值 score <= 100，整体值为 0。
  - cmd = 'a' 时，上述允许值条件的左侧为真，不再求值 cmd == 'b'，整体值为 1。

练习：比较与逻辑运算

- 题 A：写出“整数 n 在 10～20 内（含端点）”的条件；当 n = 25 时，判断各比较及整体结果。
- 题 B：cmd = 'b'，判断 cmd == 'a' || cmd == 'b' 的值，以及右侧是否求值。
- 题 C：finished = 2，判断 !finished 的值。
- 题 D：分别从 int x = 4, y = 6; 开始，判断 x = y 与 x == y 的值，以及求值后的 x。
- 题 E：x 是 int 变量，分别取 0 和 3，判断 x != 0 && 10 / x > 2 的值，以及右侧是否求值。
- 各题独立；写出判断依据，不只填最终结果。

解析：比较与逻辑运算

- 题 A：n >= 10 && n <= 20。
  - n = 25 时，左侧为真、右侧为假，整体值为 0；右侧需要求值。
- 题 B：cmd == 'a' || cmd == 'b'。
  - cmd = 'b' 时，左侧为假，继续求值右侧；右侧为真，整体值为 1。
- 题 C：finished = 2 是非零值，表示真；!finished 的值为 0。
- 题 D：两种情况分别从 x = 4, y = 6 开始。
  - x = y 的值为 6，求值后 x = 6；x == y 的值为 0，求值后 x = 4。
- 题 E：x != 0 && 10 / x > 2。
  - x = 0：左侧为假，右侧不求值，整体值为 0，没有执行除法。
  - x = 3：左侧为真，右侧求值；整数商 10 / 3 为 3，3 > 2 为真，整体值为 1。

条件运算符与逗号运算符

- **条件运算符** ?: 根据条件选择其中一个表达式：条件 ? 表达式1 : 表达式2。
  - 条件非零时只求值表达式1，条件为零时只求值表达式2；未选中的表达式不求值。
  - 本讲两个备选表达式均为 int 类型，整体取得选中项的值，类型也是 int。
  - 例如 int a = 7, b = 4;，a > b ? a : b 选择较大的值：条件为真，只求值 a，结果为 7。
- **逗号运算符**依次求值两个表达式：(表达式1, 表达式2)。
  - 先完成左侧求值，再求右侧；整体取得右侧的类型和值。
  - 例如 int x = 2;，(x = 3, x + 1) 先将 3 写入 x，再算出 4；整体为 int 值 4，求值后 x = 3。
- 声明 int p = 1, q = 2; 中的逗号分隔两个声明项，不进行上述逗号运算。

练习：条件与逗号表达式

```c
int a = 3, b = 8;
int larger = a > b ? a : b;

int x = 4;
int result = (x = x + 2, x * 2);
```

分析条件

两段代码独立分析

题 A：条件表达式

选中哪一项？整个条件表达式的类型和值是什么？larger 初始化为多少？

题 B：逗号表达式

两侧分别得到什么值，按什么顺序求值？整体的类型和值是什么？result 初始化为多少，之后 x 是多少？

题 C：两种逗号

int a = 3, b = 8; 中的逗号，与初始化 result 的表达式中的逗号，各承担什么作用？

解析：条件与逗号表达式

- 题 A：a = 3, b = 8 时，a > b 为假，只求值 : 后的 b。
  - 两个备选表达式均为 int 类型，整体为 int 值 8；larger 初始化为 8。
- 题 B：从 x = 4 开始分析 (x = x + 2, x * 2)。
  - 左侧先把 6 写入 x，赋值表达式的值为 6；右侧随后使用新的 x，得到 12。
  - 整体取得右侧的类型和值：int、12；result 初始化为 12，之后 x = 6。
- 题 C：int a = 3, b = 8; 的逗号分隔两个声明项；(x = x + 2, x * 2) 的逗号是运算符，规定左右两侧依次求值并取右侧结果。

本讲总结

关注项	关键规则
分组	括号、优先级与结合性决定分组，不保证一般操作数的实际求值先后。
类型与数值	整数除法舍去小数，事后转换不能恢复；商和余数可用于数量分组、拆位与重组。
字符编码	在 ASCII 编码下，英文字母可按对应位置转换大小写；字符数字与整数数值不同。
赋值与更新	= 赋值写入变量；自增、自减都更新变量，前置形式取新值，后置形式取旧值。
比较与逻辑	== 比较是否相等；比较与逻辑结果为 int 值 0 或 1。逻辑与的左侧为假、逻辑或的左侧为真时，不求值右侧。
条件与逗号	?: 只求值所选项；逗号运算符先求左侧，再求右侧，取得右侧的类型和值。

表达式描述计算，赋值把结果写入对象；分别记录表达式的类型和值、求值后变量的值。更新与条件将支持后续重复和选择；计算偏差仍按预期与实际逐步定位。

教材阅读指引

《C Primer Plus》第六版位置	回看与查用的主题
第 5 章 5.2～5.5 节	基本运算、表达式与语句、商余数、赋值、自增自减和类型转换
第 3 章 3.4.3 节	字符与整数编码值；结合本讲的 ASCII 条件阅读
第 6 章 6.3、6.6～6.7 节	比较、复合赋值和逗号表达式
第 7 章 7.3、7.5 节	逻辑运算、短路和条件运算符

- 按表中主题阅读对应小节；先回看本讲示例，再查用教材中的规则与解释。