顺序结构程序设计

从数据类型到格式化输出

HGNU-ACM 实验室算法组

截图配套：10 道例题，33 道练习题





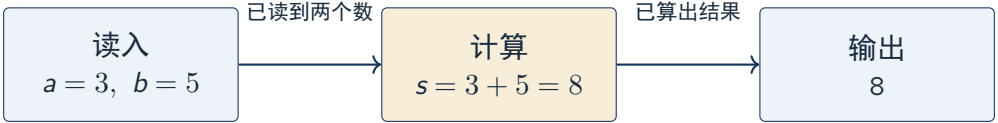
本章目录

先打牢类型、运算、转换和输入输出的基础，再进入例题。

本章目录（续）

最后做实例建模，并按练习索引逐题提交学校 OJ。

顺序执行：结果沿着三步往后传



本页要点

每步只能用已准备好的数据。若把计算行放在读入行之前，即使后来把数据读对，也不会自动重算。

输入、处理、输出 | 跟读每一行

语句	运行前要有	用 3 5 验证
`cin >> a >> b;`	已声明的变量	$a = 3,\ b = 5$
`int sum = a + b;`	两次读取后的值	$sum = 8$
`cout << sum;`	已完成的计算	屏幕显示 8

只改输入，不改程序

再试 `-3 3`，预期输出 0。若只交换两个输入的顺序，和是否改变？

完整演示：`code/sequence.cpp`。课件的短片段省略了换行，源码会在末尾换行。

【基础】语句、分号与缩进

- 一条语句以分号 ；结束；漏写分号是最常见的编译错误之一。
- 花括号 { } 表示一个语句块；本课程统一让左花括号单独换行。
- 缩进统一用 4 个空格，层次一目了然；不要把空格和 Tab 混在一起。
- 注释用 //（到行尾）或 /* ... */（可跨行），注释不参与编译。

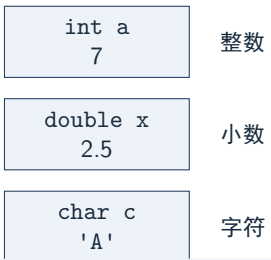
```
int a = 3;              // 声明并初始化
a = a + 1;              // 赋值：a 变成 4
cout << a << '\n';      // 输出 4 并换行
```

本页要点

先保证每条语句独立成行、缩进一致，再追求写得短。

【基础】变量：名字、类型、值

- 变量是内存中的一块空间，有名字、类型和当前值。
- 声明 `int a;` 告诉编译器“我要一个整数”。
- 初始化 `int a = 7;` 在声明的同时放入初值。
- 未初始化的局部变量值是随机的，使用前必须赋值。




```
int a = 7;          // 整数
double x = 2.5;     // 小数
char c = 'A';       // 单个字符
```

本页要点

类型决定这块内存能存什么、占多大、能做哪些运算。

选类型：先圈出可能超限的那一步

设三笔人数分别为 10^9。纸上按计算顺序相加：

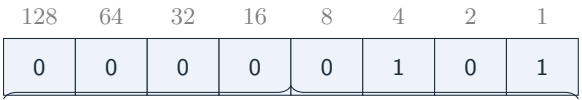
步骤	实际数学值	常见的 32 位 `int` 能存？
每个输入	$1\,000\,000\,000$	能
第一次加法	$2\,000\,000\,000$	能
第二次加法	$3\,000\,000\,000$	不能，上限约为 2.147×10^9

本页要点

把输入变量声明为 `long long`，使加法本身在长整数中执行。只把接收结果的变量改大，前面的加法仍可能溢出。

【基础 2.1】计算机只认识 0 和 1

- 内存由大量二进制位（bit）组成，每一位只能是 0 或 1。
- 8 位组成 1 个字节（byte）；类型决定用几个字节、如何解释这些位。
- 同一个二进制串，按 `int` 解释是一个整数，按 `char` 解释是一个字符。

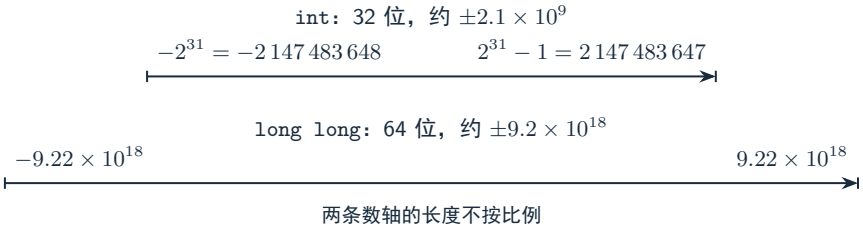


$00000101_2 = 5$

本页要点

类型不是“多此一举”，它决定编译器怎样解释同一串 0/1。

【基础 2.2】int 与 long long：能存多大



- `int`：4 字节，日常整数与 10^9 级数据。
- `long long`：8 字节，和、乘积可能超过 2.1×10^9 时使用。

```
int a = 2000000000;
long long b = 1LL * a * 2;    // 先转 long long 再乘
```

本页要点

读题先看数据范围；中间结果比最终答案更容易溢出。

【基础 2.3】溢出：答案为什么会变成负数

```
int a = 2147483647;    // int 的最大值
a = a + 1;             // 溢出，结果不可靠
cout << a << '\n';

long long b = 2147483647;
b = b + 1;             // long long 安全
cout << b << '\n';
```

- 有符号整数溢出在 C++ 中是未定义行为，不要依赖“变成负数”这种结果。
- 把其中一个操作数提前变成 `long long`：`1LL * a * b`。
- 常量也可以写 `1000000000LL`。



有符号 int 的合法范围

本页要点

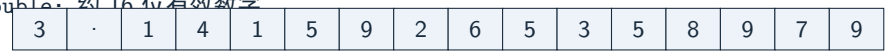
估计最大值时要把“运算过程中”的每一步都算进去。

【基础 2.4】float 与 double：小数与有效数字

float：约 7 位有效数字



double：约 16 位有效数字



- `float`：4 字节，约 $\pm 3.4 \times 10^{38}$，$6 \sim 7$ 位有效数字。
- `double`：8 字节，约 $\pm 1.8 \times 10^{308}$，$15 \sim 16$ 位有效数字。

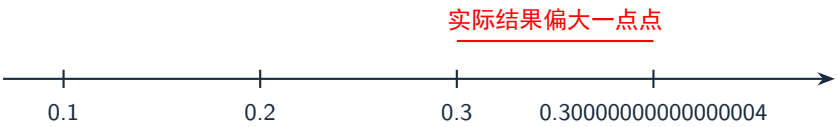
```
float f = 3.14159265f;  // 末尾 f
double d = 3.14159265;  // 默认 double
```

本页要点

题目要求小数时优先用 `double`；`float` 的精度常常不够。

【基础 2.5】0.1 + 0.2 为什么不等于 0.3

- 十进制小数 0.1 转成二进制是无限循环小数，`double` 只能存近似值。
- 因此 `0.1 + 0.2` 可能得到 `0.30000000000000004`。
- 判断两个实数相等，要看差值是否小于允许误差：`fabs(a - b) < 1e-9`。
- 输出时按题目要求保留位数，计算过程中不要提前四舍五入。

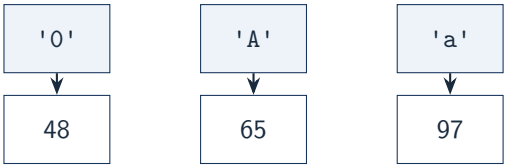


本页要点

浮点数适合“算答案”，不适合“判相等”。

【基础 2.6】char：一个字符就是一个整数编码

- char 存单个字符，用单引号：'A'、'0'、'\n'。
- 字符按 ASCII 编码存成整数：'0'=48，'A'=65，'a'=97。
- '0' 和整数 0 不是一回事。



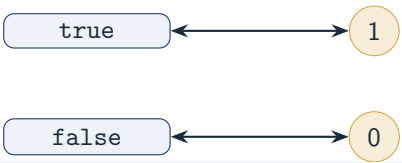

```
char c = 'A';
cout << c << '\n';          // 输出 A
cout << int(c) << '\n';     // 输出 65
cout << char(66) << '\n';   // 输出 B
```

本页要点

字符可以参与整数运算：'8' - '0' 得到数字 8。

【基础 2.7】bool：只有真和假

- bool 只有两个取值：true 和 false。
- 输出时默认打印 1 和 0；加 boolalpha 后打印 true/false。
- 非零值转成 bool 为真，0 为假。




```cpp
bool ok = true;
cout << ok << '\n';                  // 1
cout << boolalpha << ok << '\n';     // true
bool big = 5 > 3;                    // 比较的结果是 bool
```

本页要点

下一章的 if 条件，本质上就是判断一个 bool 表达式。

【基础 2.8】string：一串字符




```
string s = "HGNU";
s.length() == 4
s[1] == 'G'
```

- 需要 `#include <string>`；用双引号表示字符串。
- `cin >> s` 读到空白为止；读整行用 `getline(cin, s)`。
- 下标从 0 开始，`s[0]` 是第一个字符。

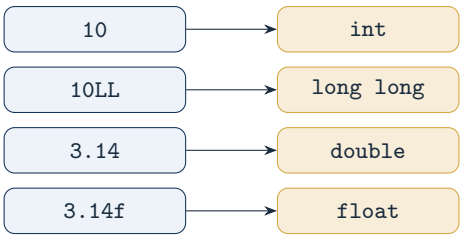
```
string s = "HG";
s = s + "NU";          // 拼接
cout << s << '\n';     // HGNU
cout << s.size();      // 4
```

本页要点

`string` 比 `char` 多了一整套字符串操作，本章先会读、拼、取长度。

【基础 2.9】常量、字面量与后缀

- 常量用 const 定义，之后不能修改。
- 整数字面量默认 int；加 LL 变 long long。
- 小数字面量默认 double；加 f 变 float。
- 1e9 是 double，不是整数。

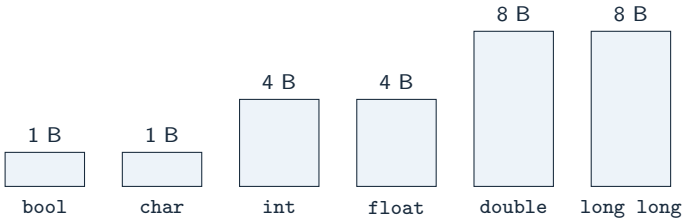



```cpp
const double PI = 3.141592653589793;  // 常量
long long big = 1000000000LL;         // 明确 64 位
double x = 1e9;                       // 1e9 是 double
```

本页要点

写常量时把“类型”和“值”一起想清楚，避免隐式转换带来的意外。

【基础 2.10】sizeof：每种类型占几个字节



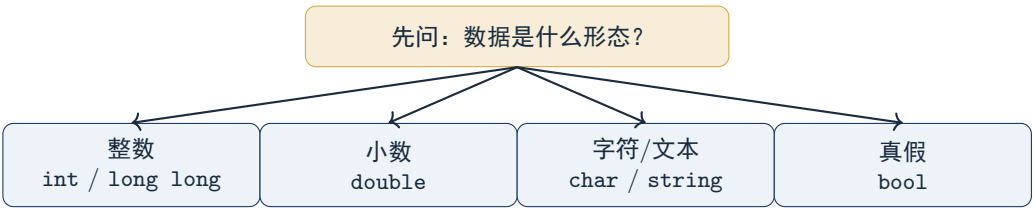

```
cout << sizeof(int) << '\n';        // 4
cout << sizeof(long long) << '\n';  // 8
cout << sizeof(bool) << '\n';       // 1
```

- sizeof 的结果类型是 size_t。
- 竞赛环境基本是表中这些值。

本页要点

类型的大小决定了它能表示的范围，也决定了会不会溢出。

【基础 2.11】怎样为题目选类型



- 整数先看范围：超过 2.1×10^9 用 `long long`。
- 小数一律先用 `double`，`float` 精度不够。
- 单个字符用 `char`，一串字符用 `string`。

本页要点

先定形态，再定范围，最后才写变量声明。

【基础 2.12】类型速查表

类型	常见大小	取值范围	典型用途
`int`	4 字节	-2^{31} 到 $2^{31}-1$	一般整数
`long long`	8 字节	-2^{63} 到 $2^{63}-1$	大整数、乘积
`float`	4 字节	约 $\pm 3.4 \times 10^{38}$	精度要求低的小数
`double`	8 字节	约 $\pm 1.8 \times 10^{308}$	一般小数
`char`	1 字节	ASCII 编码	单个字符
`bool`	1 字节	`true` / `false`	真假判断
`string`	不固定	任意长度字符串	文本

记忆方法

整数看范围，小数看精度，字符看引号，文本看长度。

用商删位，用余数取位

整数除法截去小数部分。余数是被除数扣除整份除数后留下的数。

$$\boxed{407} \longrightarrow \boxed{407/10 = 40} \longrightarrow \boxed{40 \bmod 10 = 0}$$

关键行：先用除法丢弃个位，再用对 10 取余拿十位。加括号可明确先做什么。

适用与区别：这里按非负十进制数拆位；直接用浮点除法不会截掉个位。

【基础 3.1】算术运算符：加、减、乘、除、取余

运算符	含义
`+`	加
`-`	减
`*`	乘
`/`	除（整数除法见下页）
`%`	取余（只能用于整数）

```
int a = 17, b = 5;
cout << a + b << '\n';   // 22
cout << a - b << '\n';   // 12
cout << a * b << '\n';   // 85
cout << a / b << '\n';   // 3
cout << a % b << '\n';   // 2
```



3 组，每组 5 个

余 2 个

$$17 = 5 \times 3 + 2$$

本页要点

两个整数相除，小数部分被直接丢掉，不是四舍五入。

【基础 3.2】整除与取模：商和余数

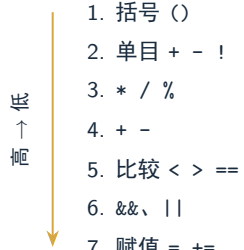
- 恒等式：`a == (a / b) * b + a % b`。
- `17 / 5 == 3`，`17 % 5 == 2`；商向零截断。
- 负数：`-17 / 5 == -3`，`-17 % 5 == -2`。
- `%` 两边必须是整数；小数取余用 `fmod`。



本页要点

拆位、判断奇偶、循环取余，都靠 `/` 和 `%`。

【基础 3.3】运算符优先级：先乘除，后加减



- 同一优先级从左到右：100 / 10 / 2 等于 (100/10)/2。
- 赋值从右到左：a = b = 5 先把 5 给 b，再给 a。
- 不确定就加括号，括号最清楚。

```
int a = 2 + 3 * 4;        // 14，不是 20
int b = (2 + 3) * 4;      // 20
int c = 100 / 10 / 2;     // 5，从左到右
```

本页要点

优先级记不住时，用括号把意图写出来，既对又易读。

【基础 3.4】把数学式翻译成表达式

数学	C++
$(a+b)/2$	`(a + b) / 2`
$a/(b+c)$	`a / (b + c)`
$2\pi r$	`2 * PI * r`
a^2	`a * a` 或 `pow(a, 2)`
$\sqrt{x}$	`sqrt(x)`

- 分数线自带括号；写成一行时必须补括号。
- 乘号不能省略：2a 要写 `2 * a`。
- `^` 在 C++ 中是按位异或，不是乘方。

```
double average = (a + b) / 2.0;   // 2.0 保证浮点除法
double area = PI * r * r;         // 不要写 PI * r^2
```

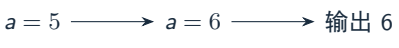
本页要点

先照数学式写，再逐项补上乘号、括号和小数点。

【基础 3.5】自增自减：++ 与 −

- a++：先使用 a 的值，再让 a 加 1。
- ++a：先让 a 加 1，再使用新值。
- 单独写成一条语句时两者一样；不要塞进复杂表达式。






```
int a = 5;
cout << a++ << '\n';   // 5，之后 a 变 6
cout << ++a << '\n';   // 7
a += 3;                // 等价于 a = a + 3
```

本页要点

初学阶段：自增自减单独成句，别和输出、赋值混在一行。

【基础 3.6】复合赋值与常见简写

写法	等价于
`a += 3`	`a = a + 3`
`a -= 3`	`a = a - 3`
`a *= 3`	`a = a * 3`
`a /= 3`	`a = a / 3`
`a %= 3`	`a = a % 3`

- 复合赋值先算右边，再和左边的旧值运算。
- `a /= 3` 仍是整数除法，小数部分会丢。
- 累加、累乘时常用，代码更短也更清楚。

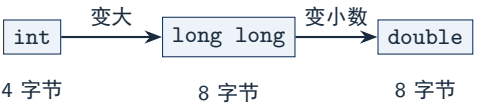
```
long long sum = 0;
sum += 10;          // 10
sum += 20;          // 30
sum *= 2;           // 60
```

本页要点

复合赋值不是“新运算”，只是“读旧值、算、写回”的缩写。

【基础 3.7】表达式里的类型：谁把谁“带上来”

- int 与 long long 运算，结果按 long long 算。
- int 与 double 运算，结果按 double 算。
- 两个 int 相乘，先按 int 算，再赋值就晚了。




```
int a = 100000, b = 100000;
long long wrong = a * b;          // 先 int 溢出
long long right = 1LL * a * b;    // 先转成 long long
```

本页要点

溢出发生在“算”的时候，不是“存”的时候。

转换要发生在运算之前

写法	除法先算出什么	存进 double 后
`double a = 5 / 2;`	整数 2	2.0
`double b = 5.0 / 2;`	小数 2.5	2.5

只改变显示会怎样？

`fixed` 与 `setprecision(1)` 只影响打印。把 2 打印成 2.0，也不会变回 2.5。

上机验证：同样的显示格式，两种除法

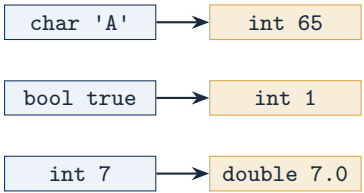
```cpp
#include <iostream>
#include <iomanip>

int main() {
    double after = 5 / 2;
    double before = 5.0 / 2;
    std::cout << std::fixed << std::setprecision(1);
    std::cout << after << '\n' << before << '\n';
}
```

运行 `code/division-demo.cpp`，应看到两行：`2.0` 和 `2.5`。再把第二个 `5.0` 改为 `5`，预测第二行。

【基础 4.1】隐式类型转换：编译器自动完成

- 小范围类型自动转成大范围类型。
- `char` 转 `int` 得到 ASCII 码；`bool` 转 `int` 得到 0/1。
- `int` 转 `double` 可能损失精度。




```
int a = 7;
double x = a;          // 7.0
char c = 'A';
int code = c;          // 65
bool ok = true;        // 1
```

本页要点

隐式转换方便，但“从大到小”时可能丢信息，下一页面讲。

【基础 4.2】double 转 int：直接截断，不四舍五入

- int(3.9) 得到 3，int(-3.9) 得到 -3，方向是“向零”。
- 赋值 int x = 3.9; 同样截断。
- 需要四舍五入时用 round(3.9)，再转成整数。



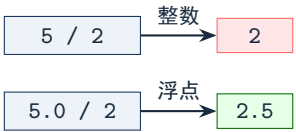

```
double x = 3.9;
int a = x;              // 3
int b = int(-3.9);      // -3
int c = round(x);       // 4
```

本页要点

题目说“四舍五入”时，不要用强制转换代替 round。

【基础 4.3】整数除法：5 / 2 为什么等于 2

- 两个操作数都是整数时，/ 执行整数除法。
- 5 / 2 == 2，5.0 / 2 == 2.5，5 / 2.0 == 2.5。
- 求平均分：先乘 1.0 或写 / 2.0。



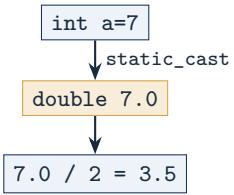

```
int a = 5, b = 2;
double x = a / b;          // 2.0，先算完再转
double y = 1.0 * a / b;    // 2.5
double z = (a + 0.0) / b;  // 2.5
```

本页要点

转换发生在运算之后，就救不回已经丢掉的小数。

【基础 4.4】显式转换：static_cast 与 C 风格

- `static_cast<double>(a)` 是推荐的写法，意图清楚。
- 老教材也写 `(double)a`，效果相同。
- 显式转换只改变这一次运算的类型，不改变变量本身。



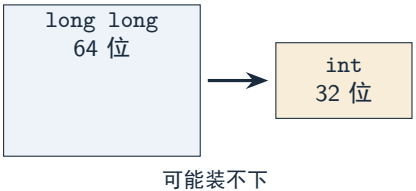

```
int a = 7, b = 2;
double x = static_cast<double>(a) / b;  // 3.5
double y = (double)a / b;               // 3.5
double z = a / b;                       // 3.0
```

本页要点

在除号左边把类型转成 `double`，整个除法就按浮点算。

【基础 4.5】大类型转小类型：可能溢出

- 把 long long 赋给 int，超出范围时结果不可依赖。
- 把负的 int 赋给 unsigned，会变成很大的正数。
- 缩小类型前，先确认值在目标范围内。




```
long long big = 3000000000LL;
int small = big;              // 可能溢出
unsigned u = -1;              // 变成大正数
```

本页要点

类型不是越大越好，但“装不下”一定出错。

【基础 4.6】三种求平均分的写法对比

写法	结果（$a=5, b=2$）	说明
`(a + b) / 2`	3	整数除法，丢小数
`(a + b) / 2.0`	3.5	2.0 是 double
`1.0 * (a + b) / 2`	3.5	先乘 1.0 提升类型

```
int a = 5, b = 2;
cout << (a + b) / 2 << '\n';         // 3
cout << (a + b) / 2.0 << '\n';       // 3.5
cout << 1.0 * (a + b) / 2 << '\n'; // 3.5
```

本页要点

看到“平均”“比例”“百分比”，先想清楚要不要小数。

输入顺序和输出格式分别核对

字段是一次显示的值及其填充空格。输入流一般跳过空白读取下个数。

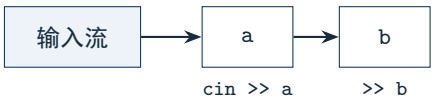
$$\boxed{\text{读}a} \longrightarrow \boxed{\text{读}b} \longrightarrow \boxed{\text{输出}a, b}$$

关键行：每个字段前分别设置宽度；固定小数和小数位数在输出数值之前设置。

适用与区别：输入的空格通常用于分隔；输出的空格和换行应按题目要求生成。

【基础 5.1】cin 与 cout：按空白分割

- `cin >> a >> b;` 跳过空格、换行，依次读入两个值。
- 程序不要打印“请输入”之类的提示语。
- `cout` 用 `<<` 连续输出．`'\n'` 换行。



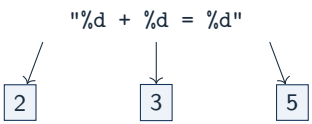

```
int a, b;
cin >> a >> b;              // 输入 3 5
cout << a + b << '\n';      // 输出 8
```

本页要点

屏幕上多打印一个字，都可能被判为格式错误。

【基础 5.2】printf 的基本格式

占位符	类型
%d	int
%lld	long long
%f	float/double
%c	char
%s	C 字符串
%%	输出一个 %



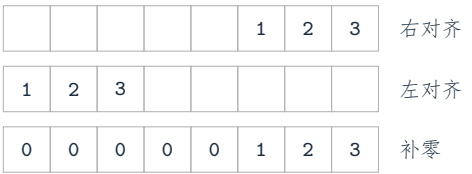

```
int a = 2, b = 3;
printf("%d + %d = %d\n", a, b, a + b);  // 三个 %d 对应三个 int
printf("%lld\n", 10000000000LL);        // long long 用 %lld
```

本页要点

占位符必须和参数类型匹配；long long 用 %lld。

【基础 5.3】printf 的宽度、精度与对齐

写法	效果
%8d	宽度 8，右对齐
%-8d	宽度 8，左对齐
%08d	宽度 8，补前导 0
%8.2f	总宽 8，保留 2 位小数
%.2f	只保留 2 位小数



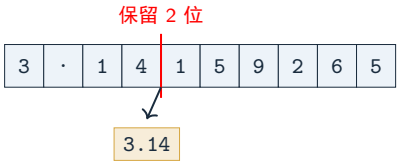

```
printf("%8d\n", 123);       // 右对齐，宽 8
printf("%-8d|\n", 123);     // 左对齐，宽 8
printf("%08d\n", 123);      // 补前导 0
printf("%8.2f\n", 3.14159); // 宽 8，保留 2 位
```

本页要点

%8d 是“至少 8 位”，数字更长时按实际宽度输出。

【基础 5.4】cout 保留小数：fixed 与 setprecision

- `fixed << setprecision(n)`：固定小数点后 n 位。
- 只写 `setprecision(n)`：控制有效数字，不是小数位。
- 设置一次后一直有效，直到再次修改。
- 输出按实际二进制值舍入，2.675 可能得到 2.67。




```
double x = 3.14159265;
cout << fixed << setprecision(2) << x << '\n';  // 3.14
cout << fixed << setprecision(4) << x << '\n';  // 3.1416
```

本页要点

题目说“保留几位小数”就用 `fixed << setprecision(n)`。

【基础 5.5】setw 与 setfill：只影响下一个字段

- `setw(5)` 只对紧随其后的一个输出项生效，下一项要重新写。
- `setfill('0')` 和 `left/right` 会一直保持。
- 多个字段时，每个字段前都写一次 `setw`。




```
cout << setw(5) << 17;        // "   17"
cout << setw(5) << 8;         // "    8"
cout << setfill('0') << setw(5) << 17;  // "00017"
```

本页要点

忘记第二个 `setw`，第二个数就不会补宽度。

【基础 5.6】输出常见错误清单

- 多打印提示语：`cout << " 请输入";` 会混进答案。
- 少打换行：多组输出粘在一行。
- 空格数量不对：值之间的空格要按题目要求。
- `long long` 用 `%d` 输出：结果错误。
- 文本里的 `'\n'` 不要多写反斜杠。

```
// 错误：多了提示语
cout << "ans=" << ans;

// 正确：只输出答案
cout << ans << '\n';
```

本页要点

提交前把输出和题面样例逐字符对照，尤其是空格和换行。

【基础 5.7】printf 与 cout 对照

需求	printf	cout
整数	%d	cout << x
长整数	%lld	cout << x
保留 2 位	%.2f	fixed << setprecision(2)
宽度 5	%5d	setw(5)
补零	%05d	setfill('0') << setw(5)
换行	\n	'\n'

怎么选

本章两种都见得到；初学建议先用 cout，遇到固定宽度或保留位数时再对照使用。

先验定义域，再调用函数

定义域是一种运算允许的输入范围。实数平方根的参数不能小于零。

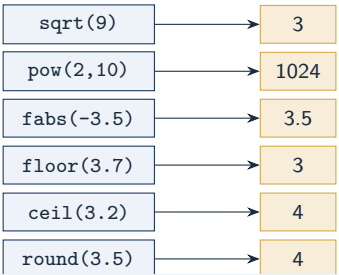
$$\boxed{a, b, c} \longrightarrow \boxed{p = (a + b + c)/2} \longrightarrow \boxed{\sqrt{p(p-a)(p-b)(p-c)}}$$

关键行：先用名为 p 的变量保存半周长，下一行才计算根号内的乘积。

适用与区别：平方根只求非负根；幂函数的浮点结果不宜直接作为精确的整数判定。

【基础 6.1】cmath：常用数学函数

函数	作用
`sqrt(x)`	平方根
`pow(x, y)`	x^y
`fabs(x)`	绝对值（浮点）
`floor/ceil`	取整（round 四舍五入）
`fmod(x, y)`	浮点取余



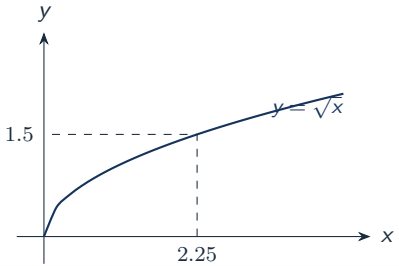

```
double a = sqrt(9);        // 3
double b = pow(2, 10);     // 1024
double c = fabs(-3.5);     // 3.5
```

本页要点

这些函数都在 <cmath> 中，接收和返回的大多是 double。

【基础 6.2】sqrt、pow、fabs 的细节

- `sqrt` 要求 $x \geq 0$，否则得到 NaN。
- `pow` 返回 `double`；平方写 `x * x`。
- `fabs` 用于 `double`，整数用 `abs`。



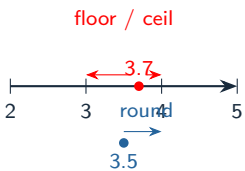

```
cout << sqrt(2.25) << '\n';    // 1.5
cout << pow(2, 10) << '\n';    // 1024
cout << fabs(-3.5) << '\n';    // 3.5
```

本页要点

函数名区分大小写，`Sqrt`、`Pow` 都不存在。

【基础 6.3】floor、ceil、round、fmod

- `floor(3.7)=3`，`ceil(3.2)=4`。
- `round(3.5)=4`，`round(-3.5)=-4`。
- `fmod(7.5, 2)=1.5`，用于浮点取余。
- 返回值都是 `double`，需要整数时再转。



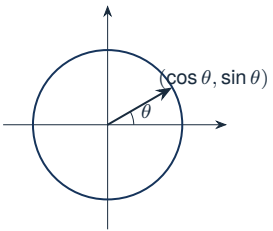

```
cout << floor(3.7) << '\n';   // 3
cout << ceil(3.2) << '\n';    // 4
cout << round(3.5) << '\n';   // 4
cout << fmod(7.5, 2) << '\n'; // 1.5
```

本页要点

“向下”是向数轴负方向，“向零”才是截断，两者对负数不同。

【基础 6.4】三角函数与圆周率

- sin、cos、tan 的参数是弧度，不是角度。
- 角度转弧度：deg * acos(-1.0) / 180。
- acos(-1.0) 是获得高精度 π 的常用写法。
- atan2(y, x) 返回点 (x, y) 的极角。




```
const double PI = acos(-1.0);   // 圆周率
double c = cos(PI / 3);         // 0.5
double d = 60 * PI / 180;       // 角度转弧度
```

本页要点

题目给角度时先转弧度，再调用 sin/cos/tan。

【基础 6.5】浮点误差与相等判断

- 浮点数有微小误差，别直接用 `a == b`。
- 判断接近：`fabs(a - b) < 1e-9`。
- 输出前不要提前四舍五入。



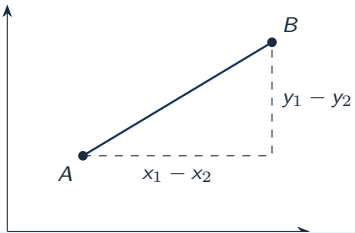

```
double a = 0.1 + 0.2, b = 0.3;
if (fabs(a - b) < 1e-9) {
    cout << "相等" << '\n';   // 实际会执行
}
```

本页要点

实数比较看“足够接近”，不要看“完全相等”。

【基础 6.6】应用：两点间距离

- 公式：$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$。
- 先算差、再平方、求和、开方，每一步都用 `double`。
- 输出保留位数由题目决定。




```cpp
double dx = x1 - x2;                    // 横向差
double dy = y1 - y2;                    // 纵向差
double dist = sqrt(dx * dx + dy * dy);  // 勾股定理
cout << fixed << setprecision(3) << dist << '\n';
```

本页要点

先写公式，再逐项翻译；不要漏掉平方和开方。

例题索引（1/2）

点击题名可跳到讲解页。

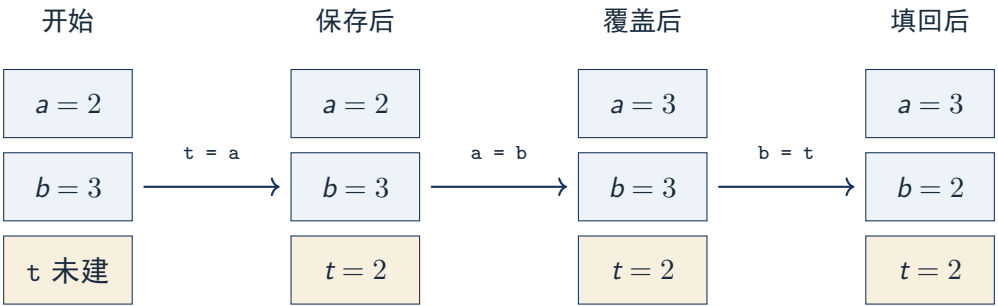
- 【例 2.1】交换值
- 【例 2.2】整数的和
- 【例 2.3】买图书
- 【例 2.5】圆
- 【例 2.6】鸡兔同笼
- 【例 2.12】糖果游戏
- 【例 2.13】数字对调

例题索引（2/2）

点击题名可跳到讲解页。

- 【例 2.14】平均分
- 【例 2.15】歌手大奖赛
- 【例 2.16】三角形面积

【例 2.1】交换值｜哪一个旧值会丢？



输入两个正整数并交换它们的值。先做 `a = b` 会丢掉原来的 2，所以第一步要先暂存。

【例 2.1】交换值｜核心步骤

模板变化：处理过程要保住即将覆盖的旧值。

核心步骤：先把 a 存入临时变量，再用 b 覆盖 a，最后把暂存值交给 b。

完整程序：`code/ex2-1.cpp`。

【例 2.2】整数的和｜题意与推导

题目

读入三个整数 a, b, c，求它们的和。

1. 三个整数按输入顺序放进三个变量。
2. 先写出结果 $s = a + b + c$；三数之和可能超过 `int` 的范围，使用 `long long`。

样例推导

$s = 1 + 2 + 3 = 6$。

【例 2.2】整数的和 | 核心步骤

模板变化：模板仍是读数、加法、输出；中间和可能超过 int 的范围。

核心步骤：把三个输入都声明为长整数，让加法从第一步起使用长整数。

完整程序：code/ex2-2.cpp。

【例 2.3】买图书 | 题意与推导

题目

小明有 n 元，一本书原价 m 元，现按八折出售。求买书后剩余的钱，保留两位小数。

1. 八折就是支付原价的 80%，不是减去 8 元。
2. 先算折后价 $0.8m$，再算余额 $n - 0.8m$；用浮点数并控制小数位。

样例推导

$n - 0.8m = 100 - 0.8 \times 100 = 20.00$。

【例 2.3】买图书 | 核心步骤

模板变化：处理过程有实数折扣，输出需固定两位。

核心步骤：先算原价的 0.8 倍，再用持有金额减去折后价。

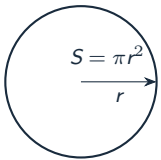
完整程序：`code/ex2-3.cpp`。

【例 2.5】圆｜题意与推导

题目

输入圆的半径 r，依次输出直径、周长、面积，每个数保留四位小数。

1. 圆周率用 $\pi = \texttt{acos(-1.0)}$，不要只取 3.14。
2. 直径 $2r$、周长 $2\pi r$、面积 πr^2。
3. 三项输出使用同一个格式控制。



样例推导

$r = 1 \Rightarrow (2r, 2\pi r, \pi r^2) = (2, 6.2832, 3.1416)$。

【例 2.5】圆｜核心步骤

模板变化：读一个半径，按题意输出三个量。

核心步骤：依次计算 $2r, 2\pi r, \pi r^2$，统一保留四位小数。

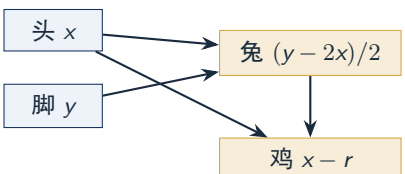
完整程序：`code/ex2-5.cpp`。

【例 2.6】鸡兔同笼｜题意与推导

题目

同一个笼子里有鸡和兔，共有 x 个头、y 只脚。求鸡和兔各多少只。

1. 设鸡 c 只、兔 r 只：头数给出 $c + r = x$。
2. 脚数给出 $2c + 4r = y$；第二式减去第一式的两倍，得 $2r = y - 2x$。



样例推导

$r = (y - 2x)/2$，$c = x - r$；样例得到 $r = c = 15$。

【例 2.6】鸡兔同笼｜核心步骤

模板变化：先假设所有头都属于鸡，再用多出的腿反推兔数。

核心步骤：每把一只鸡换成兔会多两条腿；先算兔数，再用总头数减兔数。

完整程序：`code/ex2-6.cpp`。

【例 2.12】糖果游戏 | 题意与推导

题目

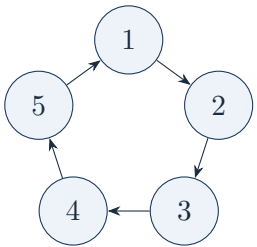
五个小朋友按 $1 \to 2 \to 3 \to 4 \to 5$ 围坐。轮到某人时，他把当前糖果分成三等份，余数吃掉；一份留给自己，另两份各给左右邻居。依次完成一轮，输出五人最终糖果数，每项宽度至少 5，右对齐。

1. 顺序很重要：2 号分糖时，他已经收到 1 号给的糖。
2. 每轮先取 $q = a_i/3$，将 a_i 改为 q，左右邻居各加 q；余数自然消失。

样例推导

样例依次处理后的最终状态为 $(11, 7, 9, 11, 6)$。

例 2.12 | 为什么必须按 $1 \to 5$ 顺序？



样例前两步

初始 $(8, 9, 10, 11, 12)$。1 号先分出 2 给左右，变为 $(2, 11, 10, 11, 14)$；随后 2 号按当前的 11 分，不能再用初始的 9。

本页要点

五个人的状态互相影响，必须严格按照 $1 \to 2 \to 3 \to 4 \to 5$ 的顺序更新。

【例 2.12】糖果游戏 | 核心步骤

模板变化：数据会被前面的人改变，按题目顺序就地更新同一张表。

核心步骤：轮到某人时以当前糖果数整除 3 得一份；自己留一份，两份分别发给左右邻居。

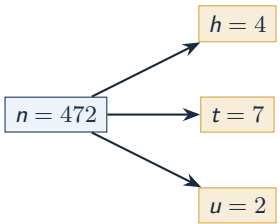
完整程序：`code/ex2-12.cpp`。

【例 2.13】数字对调 | 题意与推导

题目

输入一个三位正整数，交换百位与个位后输出新整数。

1. 百位 $h = n/100$，十位 $t = (n/10)\%10$，个位 $u = n\%10$。
2. 重组顺序为个位、十位、百位：$100u + 10t + h$。



样例推导

$123 \to (h, t, u) = (1, 2, 3) \to 321$。

【例 2.13】数字对调｜核心步骤

模板变化：用商和余数拆开十进制数字，再按新权重相加。

核心步骤：取百位、十位和个位；让个位乘 100、十位乘 10、百位乘 1。

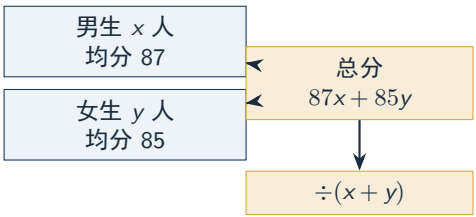
完整程序：`code/ex2-13.cpp`。

【例 2.14】平均分 | 题意与推导

题目

男生 x 人，平均分 87；女生 y 人，平均分 85。求全班平均分，保留四位小数。

1. 不能把 87 与 85 直接取平均，除非男女生人数相等。
2. 先求总分 $87x+85y$，再除以总人数 $x+y$。



样例推导

$\bar{s}=(87x+85y)/(x+y)=(174+255)/5=85.8$。

【例 2.14】平均分 | 核心步骤

模板变化：两组人数可能不同，不能直接平均两个组均分。

核心步骤：先用各组人数乘本组均分并求和，再除以总人数；让除法使用浮点数。

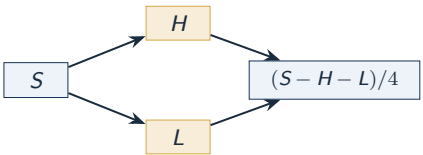
完整程序：`code/ex2-14.cpp`。

【例 2.15】歌手大奖赛｜题意与推导

题目

六位评委的平均分为 9.6；去掉一个最高分后，剩余五人平均分为 9.4；去掉一个最低分后，剩余五人平均分为 9.8。求最高分和最低分都去掉后的四人平均分。

1. 六人的总分 $S = 6 \times 9.6$。
2. 最高分 $H = S - 5 \times 9.4$；最低分 $L = S - 5 \times 9.8$。
3. 最后四人的平均分是 $(S - H - L)/4$。



样例推导

$S = 57.6$，$H = 10.6$，$L = 8.6$，故 $(57.6 - 10.6 - 8.6)/4 = 9.60$。

【例 2.15】歌手大奖赛 | 核心步骤

模板变化：反推出最高、最低分后再做去极值平均。

核心步骤：用六人总分减去对应五人总分，分别得到两个极值；扣除它们后除以 4。

完整程序：`code/ex2-15.cpp`。

【例 2.16】三角形面积｜题意与推导

题目

已知三角形三边 a, b, c，用海伦公式计算面积，保留三位小数。令半周长 $p = (a + b + c)/2$，则面积 $S = \sqrt{p(p-a)(p-b)(p-c)}$。

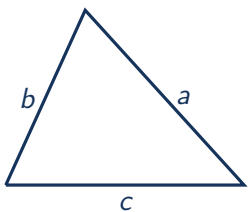
1. 先算半周长 $p = (a + b + c)/2$，再代入海伦公式。
2. 计算平方根需要 `sqrt`，定义在 `<cmath>` 中。
3. 保留三位小数用 `fixed << setprecision(3)`。

样例推导

$p = 4.65$；$S = \sqrt{4.65(4.65-2.1)(4.65-3.1)(4.65-4.1)} \approx 3.179$。

例 2.16 | 公式中的每个量

- 三边 a, b, c 必须先满足三角形不等式。
- 半周长 $p = (a + b + c)/2$。
- 海伦公式：$S = \sqrt{p(p-a)(p-b)(p-c)}$。
- `sqrt` 来自 `<cmath>`；最终保留 3 位小数。



【例 2.16】三角形面积｜核心步骤

模板变化：处理分两行，后一行使用先求出的半周长。

核心步骤：令 $p=(a+b+c)/2$，再求 $\sqrt{p(p-a)(p-b)(p-c)}$；输入边长需能组成三角形。

完整程序：`code/ex2-16.cpp`。

练习顺序：按截图的五节推进

- 第一节：运算符和表达式；第二节：常量和变量。
- 第三节：标准数据类型；第四节：数据输入输出。
- 第五节：顺序结构实例，综合使用前面的知识。

完成标准

每题先独立写出输入、计算、输出三步；提交后根据评测结果检查式子、类型和格式。

本章练习题索引 | 1–9 / 33

1. P5007 | 计算 (a+b)×c 的值（一本通 1007）
2. P5008 | 计算 (a+b)/c 的值（一本通 1008）
3. P5009 | 带余除法（一本通 1009）
4. P5010 | 计算分数的浮点数值（一本通 1010）
5. P5011 | 甲流疫情死亡率（一本通 1011）
6. P5012 | 计算多项式的值（一本通 1012）
7. P5013 | 温度表达转化（一本通 1013）
8. P5014 | 与圆相关的计算（一本通 1014）
9. P5015 | 计算并联电阻的阻值（一本通 1015）

点击蓝色题名进入学校 OJ；题号已逐一核对，不使用一本通直链。

1. P20176 | 成绩（一本通 1414）
2. P5016 | 整型数据类型存储空间大小（一本通 1016）
3. P5017 | 浮点型数据类型存储空间大小（一本通 1017）
4. P5018 | 其他数据类型存储空间大小（一本通 1018）
5. P5019 | 浮点数向零舍入（一本通 1019）
6. P5020 | 打印 ASCII 码（一本通 1020）
7. P5021 | 打印字符（一本通 1021）
8. P5022 | 整型与布尔型的转换（一本通 1022）
9. P5023 | Hello,World! 的大小（一本通 1023）

点击蓝色题名进入学校 OJ；题号已逐一核对，不使用一本通直链。

1. P5024 | 保留 3 位小数的浮点数（一本通 1024）
2. P5025 | 保留 12 位小数的浮点数（一本通 1025）
3. P5026 | 空格分隔输出（一本通 1026）
4. P5027 | 输出浮点数（一本通 1027）
5. P5028 | 字符菱形（一本通 1028）
6. P5029 | 计算浮点数相除的余（一本通 1029）
7. P5030 | 计算球的体积（一本通 1030）
8. P5031 | 反向输出一个三位数（一本通 1031）
9. P5032 | 大象喝水查（一本通 1032）

点击蓝色题名进入学校 OJ；题号已逐一核对，不使用一本通直链。

本章练习题索引 | 28–33 / 33

1. P5033 | 计算线段长度（一本通 1033）
2. P5034 | 计算三角形面积（一本通 1034）
3. P5035 | 等差数列末项计算（一本通 1035）
4. P5036 | A×B 问题（一本通 1036）
5. P5037 | 计算 2 的幂（一本通 1037）
6. P5038 | 苹果和虫子（一本通 1038）

点击蓝色题名进入学校 OJ；题号已逐一核对，不使用一本通直链。

提交前检查清单

1. 式子：变量类型、括号、整除与取模是否正确。
2. 输入：读入顺序、数量、多组数据是否处理。
3. 输出：空格、换行、小数位数、宽度是否与题面一致。
4. 边界：0、负数、最大值、最小值是否手算验证。
5. 提交：只提交源代码，不输出提示语。

刷题建议

每题先在纸上写“输入—计算—输出”三步，再写代码；错题按“看错题、算错、写错、格式错”记录原因。