C++ 语言入门

从第一行程序到读题、计算与输出

HGNU-ACM 实验室算法组

配套：截图中的 4 道例题 $+$ 4 道练习题





本章目录

起步：程序、编译与运行

输出：cout 与格式控制

变量、基本类型与输入

前三节是零基础必学：程序骨架、编译运行、输出输入与变量。

本章目录（续）

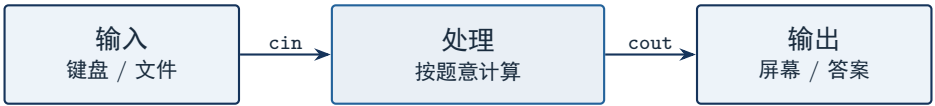
四道例题精讲

四道练习逐题讲

检查与课后安排

基础打牢后，再用 4 道例题和 4 道练习完成从读题到提交的闭环。

程序在做什么？

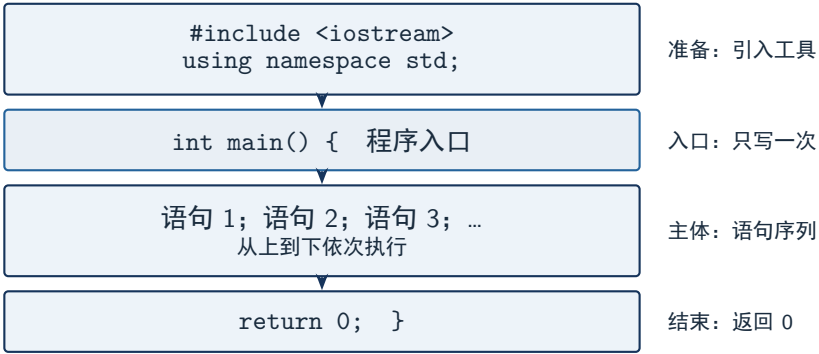


- 输入：题目给的数据；有些题没有输入。
- 处理：按题意计算，或决定要打印什么。
- 输出：只写答案，空格、换行、小数位都要与要求一致。

本页要点

写程序前先回答三个问题：输入是什么？要算什么？输出长什么样？

程序的基本结构：四层骨架



C++ 程序由头文件、主函数和语句组成；所有语句都必须写在函数体内部。

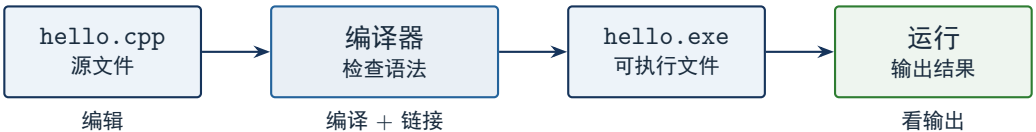
骨架逐行拆解

```cpp
#include <iostream>      // 头文件：引入 cin、cout
using namespace std;     // 命名空间：直接写 cout

int main() {             // 主函数：程序入口
    cout << "Hello" << '\n';   // 语句：输出一行
    return 0;            // 返回 0：正常结束
}
```

- 以 # 开头的行是预处理指令，行末不加分号。
- `using namespace std;` 与普通语句一样，以分号结束。
- `main` 后的圆括号表示函数，花括号包住函数体。
- 函数体中的语句从上到下执行；`return 0;` 结束 `main`。

从源码到运行：四个阶段



- `.cpp` 只是文本，CPU 不能直接执行。
- 编译把源码翻译成机器指令；链接再把 `cout` 等库代码接上。
- OJ 点“提交”后，服务器自动完成编译、运行、比对输出。

本页要点

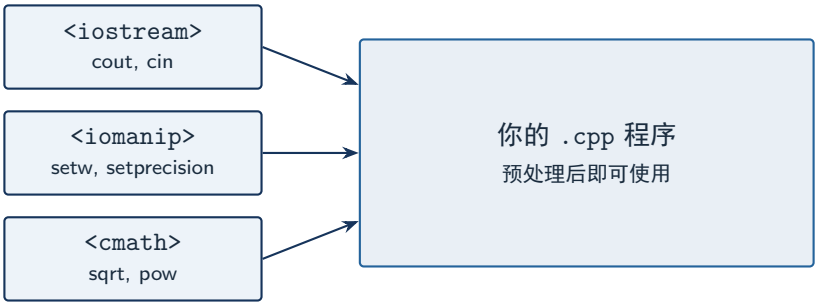
编译错误说明程序还没跑起来；答案错误才是逻辑或格式问题。

编译与运行：命令行与 IDE

```
g++ -std=c++17 hello.cpp -o hello.exe
hello.exe
Hello World!
```

- `-std=c++17` 指定语言标准；`-o hello.exe` 指定输出文件名。
- Windows 运行 `hello.exe`；Linux 与 OJ 上运行 `./hello`。
- Dev-C++、VS Code 的“编译运行”按钮做的是同一件事。
- 编译失败时不会生成新的可执行文件，先修第一条报错。

#include 与头文件：把工具请进来



- `#include` 是预处理指令：编译前把头文件内容“复制”到这一行。
- 头文件提供声明；真正的库代码在链接阶段接上。

本页要点

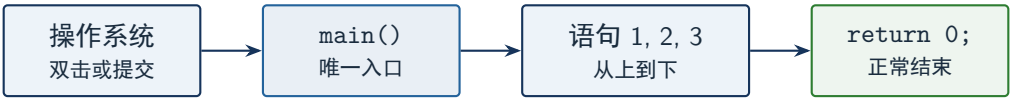
忘记包含头文件，常见报错是 `'cout' was not declared in this scope`。

常用头文件速查

头文件	提供的工具	典型用途
<iostream>	cin, cout	输入、输出
<iomanip>	setw, setprecision, fixed	控制宽度与小数位
<cmath>	sqrt, pow, fabs	数学函数（下一章）
<algorithm>	sort, max, min	排序与最值（后续）
<string>	string	字符串（后续）

- 本章最常用的是 <iostream>；需要格式控制时再加 <iomanip>。
- 头文件按需包含，不要背一大串；用哪个查哪个。

main 函数：程序的入口



- 每个程序有且只有一个 `main`，执行从 `main` 的第一条语句开始。
- `int main()` 的 `int` 表示返回整数；`return 0;` 表示正常结束。
- 右花括号 `}` 与左花括号 `{` 必须成对出现。

本页要点

`void main()` 不是标准写法，部分 OJ 会直接编译失败；统一写 `int main()`。

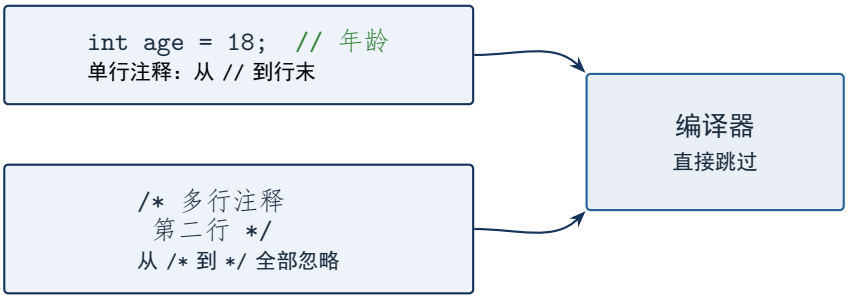
main 的写法与常见错误

```cpp
int main() {                    // 推荐写法
    cout << "OK" << '\n';
    return 0;
}
```

```cpp
void main() {                   // 不标准：返回类型应为 int
    cout << "OK";
}                               // 缺少 return 0;
```

- mian、Main 都是错的：C++ 严格区分大小写。
- 一个项目里出现两个 main，链接阶段会报重复定义。

注释：写给人的说明



- // 写单行注释；/* ... */ 写多行注释，但不能嵌套。
- 注释不参与编译，写“为什么这样做”，不要逐字翻译代码。

注释小例子与书写规范

```cpp
int main() {
    int price = 10;          // 单价：元
    int count = 3;           // 数量
    // 总价 = 单价 * 数量
    int total = price * count;
    cout << total << '\n';   // 输出 30
    return 0;
}
```

- 行尾注释对齐，代码和说明都更好读。
- 注释里可以写中文；源文件用 UTF-8 保存，避免乱码。
- 不要用注释“删代码”；提交前清理调试用的临时注释。

常见编译错误（一）：中文标点

```
int a = 3;
英文分号 ;：细而窄
```

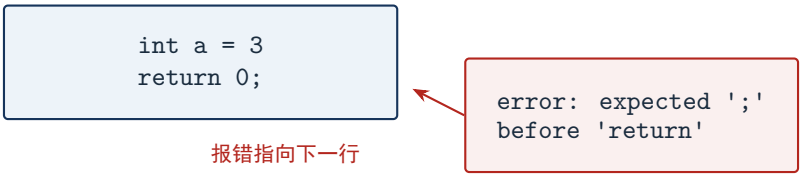
```
int a = 3，
中文逗号：宽，编译器不认识
```

- 中文 ；，（）“” 与英文 ;,()"" 外形不同，编译器只认英文。
- 写代码时把输入法切到英文标点；全角空格也会触发 stray 错误。

本页要点

报错里出现 stray 或 expected ';'，先检查上一行的标点。

常见编译错误（二）：缺分号与花括号



- 分号是语句结束标志；`int a = 3` 少了 `;`，编译器要到下一行才发现。
- 花括号必须成对；`expected '}' at end of input` 表示少了右花括号。
- 一行只写一条语句，漏分号一眼就能看出来。

读懂编译器的报错

```
hello.cpp:5:12: error: expected ';' before 'return'
    5 | return 0;
      | ^
```

- 格式是“文件名: 行号: 列号: 错误类型”，先看行号。
- 只修第一条错误；后面的报错往往是连锁反应。
- 报错行不一定真有错，可能上一行漏了分号或少了右括号。
- 在 IDE 中双击报错可跳到对应行。

本页要点

从第一条错误开始，修一处、编译一次，不要一次改十处。

第一个完整程序：逐行注释

```cpp
#include <iostream>                 // 输入输出
#include <iomanip>                  // 格式控制
using namespace std;                // 标准命名空间

int main() {
    int age = 18;                   // 整数：年龄
    double h = 1.75;                // 小数：身高
    char grade = 'A';               // 字符：等级

    cout << "age = " << age << '\n';
    cout << "height = " << fixed << setprecision(2) << h << '\n';
    cout << "grade = " << grade << '\n';
    return 0;                       // 正常结束
}
```

这份代码把本章的基础串了一遍：头文件、变量、三种类型、输出格式与返回值。

动手实验：改错三处

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

int main() {
    int a = 3;                  // 错误 1
    int b = 4                   // 错误 2
    cout << a + b << '\n'       // 错误 3
    return 0;
}
```

任务

找出三处错误并改正，编译运行；程序应输出 7。

- 提示：第 5 行用了中文分号（；），另有两行缺分号。
- 改完后把 int 改成 long long，观察是否仍能通过。

cout：把数据送到屏幕



- cout 是标准输出流，<< 把右边的值送进去。
- 可以连续写多个 <<，从左到右依次输出。
- cout 不会自动加空格或换行，格式要自己控制。

本页要点

输出是“照相机”：屏幕上看到什么，评测系统就拿到什么。

cout 的三种常见写法

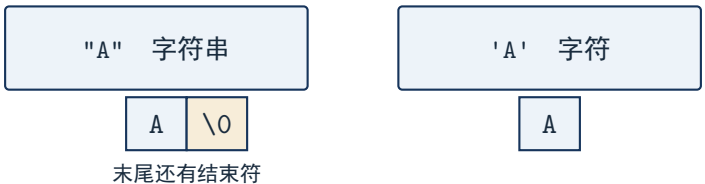
```
cout << "Hello";                    // 输出字符串
cout << 42;                         // 输出整数
cout << "A" << ' ' << 7 << '\n';    // 混合输出，末尾换行
```

- 双引号 `"..."` 是字符串，可以包含多个字符。
- 单引号 `'...'` 是字符，只能放一个字符。
- `'\n'` 是换行符；`' '` 是空格字符。

小例子

这段代码依次输出 Hello、42、A 7；最后一行在 7 后换行。

字符串与字符：双引号 vs 单引号



- `"A"` 是字符串，不能直接赋给 `char` 变量。
- `'A'` 是字符，可以赋给 `char`，输出时只打印 `A`。
- `"AB"` 不能用单引号；单引号里只能有一个字符。

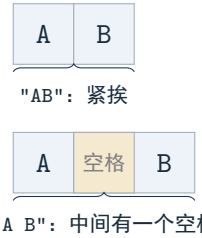
转义字符：看不见的字符

写法	含义	输出效果
'\n'	换行	光标移到下一行行首
'\t'	制表符	跳到下一个制表位
'\\'	反斜杠	输出一个 \

```
cout << "第一行\n第二行\n";    // 输出两行
cout << "A\tB\n";              // A 与 B 之间空一个制表位
cout << "反斜杠：\\" << '\n';  // 输出一个反斜杠
```

- 反斜杠 \ 是转义符号，要和后面的字符一起看。
- 在文字中表示 '\n' 时，源码里只写一个反斜杠命令。

空格与换行：把不可见变可见



- 空格也是字符，会占一个输出位置；多一个空格就是格式错误。
- 每行末尾用 '\n' 换行，不要让答案全挤在一行。

格式化输出：fixed 与 setprecision

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

int main() {
    double x = 3.14159;
    cout << fixed << setprecision(2) << x << '\n';   // 3.14
    cout << setprecision(4) << x << '\n';            // 3.1416
    return 0;
}
```

- `fixed` 表示按固定小数位输出，一旦设置就保持。
- 在 `fixed` 下，`setprecision(n)` 表示小数点后 n 位，并四舍五入。
- 不加 `fixed` 时，`setprecision` 表示有效数字位数，容易混。

setw：固定宽度与右对齐




```
int a = 7, b = 123;
cout << setw(5) << a << setw(5) << b << '\n';   // 各占 5 个字符宽
```

- `setw(n)` 为下一个输出项保留至少 n 个字符宽度。
- 默认右对齐，不足补空格；超过宽度不截断，按原样输出。
- 每个输出项都要单独写一次 `setw`。

输出检查清单

1. 有没有多打印“请输入”之类的提示语？
2. 空格、换行、大小写、英文标点是否与样例完全一致？
3. 小数位数是否正确？要求 `400.00` 时不能输出 `400`。
4. 每行末尾是否换行？最后一行也建议以 `'\n'` 结束。
5. 多组数据时是否多输出了空行或分隔线？

对照样例

把样例输出复制到记事本，打开“显示空格”逐字符比对；肉眼看不出的空格最容易被判错。

小例子：打印一张信息卡

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

int main() {
    cout << "name: Alice" << '\n';   // 第一行
    cout << "age: " << 18 << '\n';
    cout << "score: " << fixed << setprecision(1) << 92.5 << '\n';
    return 0;
}
```

运行结果

```
name: Alice
age: 18
score: 92.5
```

变量：有名字的盒子

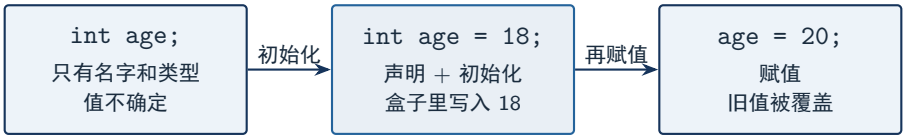


- 变量名：程序中使用的名字，如 `age`。
- 类型：决定能存什么、占多少空间，如 `int`。
- 值：盒子里当前存放的数据，可以读取，也可以改写。

本页要点

先声明、后使用；变量名区分大小写。

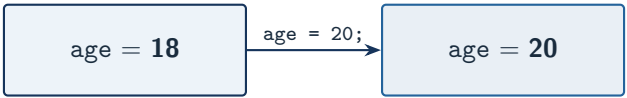
声明、初始化与赋值




```
int age;                // 声明：盒子还没有确定的值
age = 18;               // 赋值：把 18 写进盒子
int score = 90;         // 声明的同时初始化
```

局部变量不初始化时值不确定，务必在声明时给初值。

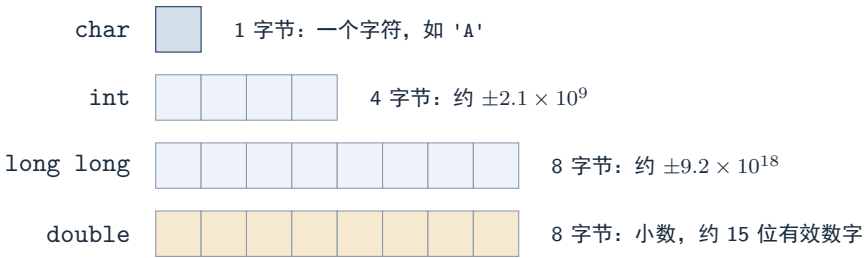
赋值是“写入”，不是“相等”




```
int a = 5;
int b = 8;
a = b;          // 把 b 的值复制给 a，a 变成 8
b = a;          // b 仍是 8，两个变量都等于 8
```

- = 是赋值；判断相等要用 ==（下一章）。
- 赋值方向从右到左：先算右边，再写入左边。

基本类型：先认识四个



- 整数优先用 `int`；结果可能超过 2×10^9 时换 `long long`。
- 有小数用 `double`；单个字符用 `char`；真/假用 `bool`。

取值范围与溢出

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

int main() {
    int a = 100000;
    int b = 100000;
    cout << a * b << '\n';          // 溢出：结果已超出 int
    long long c = 1LL * a * b;      // 先提升为 long long 再乘
    cout << c << '\n';              // 正确：10000000000
    return 0;
}
```

- `a * b` 先按 `int` 计算，溢出后再赋值已经来不及。
- `1LL *` 把整个乘法提升为 64 位；也可写 `(long long)a * b`。
- 题目范围接近 10^9 时，乘积一定要用 `long long`。

变量命名：让人看懂

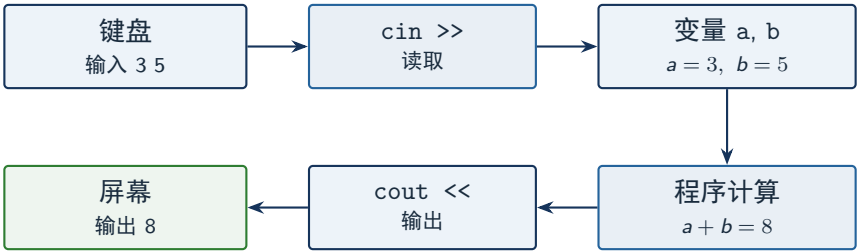
- 只能由字母、数字、下划线组成，且不能以数字开头。
- 不能使用关键字：`int`、`return`、`main` 等。
- 区分大小写：`sum` 与 `Sum` 是两个不同的变量。
- 用有意义的名字：`price`、`count`，少用 `a1`、`x2`。

合法	不合法
`int score;`	`int 2score;`
`double average;`	`int my score;`
`int max_value;`	`int return;`

本页要点

变量名对应题目中的量，代码才能和纸上的公式互相核对。

cin：从输入读进变量




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

本页要点

变量必须先声明；>> 的方向与 << 相反；OJ 上不要打印提示语。

cin 的读取规则

```
int a, b, c;
cin >> a >> b >> c;   // 依次读入三个整数
```

- 默认以空格、制表符、换行分隔，并自动跳过这些空白。
- 输入 `1 2 3`、`1` 换行 `2` 换行 `3`，效果相同。
- 类型要匹配：把 `3.14` 读进 `int` 只会得到 `3`，剩下 `.14` 可能干扰后续读取。
- 含空格的整行字符串不能用 `cin >>` 直接读，后面会学 `getline`。

小实验

在数字之间多打几个空格，输出仍然是 6。

算术运算：+ - * / %

运算	符号	例子	结果
加	+	7 + 2	9
减	-	7 - 2	5
乘	*	7 * 2	14
除	/	7 / 2	3（整数除法）
取余	%	7 % 2	1

```
int x = 2 + 3 * 4;        // 14: 先乘后加
int y = (2 + 3) * 4;      // 20: 括号优先
```

- 优先级：() 最高，其次 * / %，最后 + -；同级从左到右。
- 取余 % 只能用于整数；结果的正负与被除数相同。

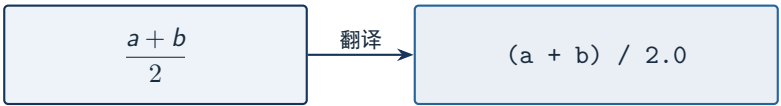
整数除法与取模




```
cout << 7 / 2 << '\n';        // 3: 小数部分舍去
cout << 7 % 2 << '\n';        // 1: 余数
cout << 7 / 2.0 << '\n';      // 3.5: 至少一边是小数
```

- 两个整数相除，结果仍是整数，向零取整（$-7/2$ 得 -3）。
- 想得到小数，至少让一边是 double，如 `7 / 2.0`。
- `%` 常用来判断整除与奇偶：`x % 2 == 0` 表示偶数。

表达式翻译：从数学到代码



- $10x$ 写成 `10 * x`，乘号不能省略。
- $\frac{a}{b}$ 写成 `1.0 * a / b`，先乘 `1.0` 避免整数除法。
- a^2 本章写成 `a * a`；`pow` 函数下一章再学。
- 先写数学式，再逐步加括号，最后对照检查一遍。

本页要点

括号不嫌多：`(a + b) / 2.0` 比 `a + b / 2.0` 安全。

小例子：摄氏度转华氏度

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

int main() {
    double c;
    cin >> c;                              // 输入摄氏度
    double f = 1.8 * c + 32;               // 换算公式
    cout << fixed << setprecision(1) << f << '\n';
    return 0;
}
```

样例

输入 25，输出 77.0；输入 0，输出 32.0。

用 double 读入，公式中的 1.8 也保证按小数计算。

复合赋值与自增（了解）

```
int x = 5;
x += 3;         // 等价于 x = x + 3，x 变成 8
x *= 2;         // 等价于 x = x * 2，x 变成 16
x++;            // 等价于 x = x + 1，x 变成 17
cout << x << '\n';
```

- 复合赋值是简写，读代码时要能还原成完整形式。
- x++ 表示加 1；本章写完整形式更清楚。
- 初学阶段不追求短，追求每一步都能解释。

输入输出全流程回顾



检查顺序

变量类型对不对？输入读全了吗？公式有没有整数除法？输出格式与样例一致吗？

这五步就是本章所有例题和练习的通用骨架。

例 1.1 | 计算机输出：先读懂要求

题目

不提供输入。打印一行 `Hello World!`。

- 输入：无。
- 处理：无计算，答案是固定文本。
- 输出：两个单词、一个空格、一个英文感叹号、最后换行。

本页要点

固定输出题也值得认真做：它能一次检查环境、编译、运行、提交全过程。

例 1.1 | 完整代码与检查

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

int main() {
    cout << "Hello World!" << '\n';    // 固定输出
    return 0;
}
```

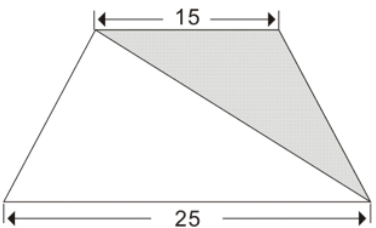
手工检查

运行后恰好一行：`Hello World!`。注意 `H` 与 `W` 均大写，中间只有一个空格。

若输出 `Hello world!` 或 `HelloWorld!`，都是答案错误。

例 1.2 | 梯形面积：先找共同的高

- 梯形上底 $a = 15$，下底 $b = 25$。
- 阴影三角形的底是上底 15，面积 $S_{\triangle} = 150$。
- 三角形的高就是梯形的高 h。



第一步

$S_{\triangle} = \dfrac{ah}{2}$，所以 $h = \dfrac{2S_{\triangle}}{a} = 20$。

例 1.2 | 再算面积，最后决定输出格式

$$
\begin{aligned}
S_{\text{梯形}} &= \frac{(a+b)h}{2} \\
&= \frac{(15+25)\times 20}{2} = 400.
\end{aligned}
$$

- 题目要求保留两位小数，所以输出 `400.00`。
- 数学结果是 400，显示格式仍是题意的一部分。
- 若题目要求四舍五入到两位，`fixed` 与 `setprecision(2)` 会自动处理。

本页要点

先算对，再按格式输出；不要用 `cout << 400` 代替 `400.00`。

例 1.2 | 代码：控制小数位数

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

int main() {
    double h = 2.0 * 150 / 15;            // 先求高
    double area = (15 + 25) * h / 2;      // 再求面积
    cout << fixed << setprecision(2) << area << '\n';
    return 0;
}
```

本页要点

`fixed` 与 `setprecision(2)` 搭配，按固定小数格式打印两位。

例 1.3 | 电影票：把故事化成式子

题目

每张儿童票 10 元，输入人数 x，输出人数和总票价。

1. 输入：一个整数 x。
2. 处理：总价 $p = 10x$。
3. 输出：x、一个空格、p，然后换行。

样例

$x = 2 \Rightarrow p = 20$，输出 `2 20`。

先确定“输入—公式—输出”三件事，再动手写代码。

例 1.3 | 完整代码

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

int main() {
    long long x;
    cin >> x;                          // 读入人数

    long long price = 10 * x;          // 总价
    cout << x << ' ' << price << '\n';
    return 0;
}
```

自测：$x = 0, 1, 2$；确认输出随输入按公式变化。

类型选择

人数可能很大，`10 * x` 用 `long long` 承接，避免 `int` 溢出。

例 1.4 | 牛吃牧草：统一单位

题意

同一片草地，15 头牛吃 20 天，或 20 头牛吃 10 天；青草每天匀速生长。

- 令 G 为原有草量，单位：一头牛一天的食量。
- 令 g 为每天新长草量，也按这个单位计。

两种情形：$G + 20g = 300$，$G + 10g = 200$。



例 1.4 | 消去初始草量

$$
\begin{aligned}
(G + 20g) - (G + 10g) &= 300 - 200, \\
10g &= 100, \\
g &= 10.
\end{aligned}
$$

- 每天新长的草，恰好够 10 头牛吃一天。
- 本题没有输入，程序可以直接计算并输出 10。

本页要点

两式相减时注意：牛数乘天数，才是总消耗量。

例 1.4 | 把推导写成程序

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

int main() {
    int used1 = 15 * 20;                    // 第一种情形总消耗
    int used2 = 20 * 10;                    // 第二种情形总消耗
    int growth = (used1 - used2) / (20 - 10);

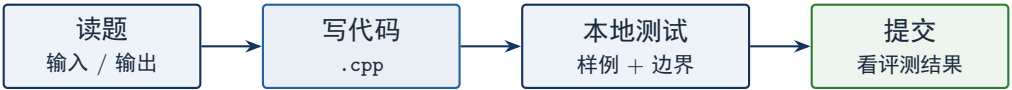
    cout << growth << '\n';                 // 输出 10
    return 0;
}
```

变量名对应含义，代码才能与纸上的方程互相核对。

检查

两个总消耗相减，再除以天数差，得到每天新增草量。

练习总览：从读题到提交



1. 学校 OJ：P5002 | 输出第二个整数：练习变量与多项输入。
2. 学校 OJ：P5003 | 对齐输出：练习固定宽度、右对齐。
3. 学校 OJ：P5004 | 字符三角形：练习字符与多行输出。
4. 学校 OJ：P5005 | 地球人口承载力估计：练习列方程与两位小数。

练习 1002 | 输出第二个整数

题目与入口

输入三个整数 a, b, c，只输出中间的 b。

学校 OJ：P5002 | 输出第二个整数

1. 先声明三个变量：`int a, b, c;`。
2. 用 `cin >> a >> b >> c;` 按顺序读取。
3. 用 `cout << b << '\n';` 输出。

样例自查

输入 `123 456 789`，输出 `456`。

本页要点

题目只要求输出 b，不要顺手把 a、c 也打印出来。

练习 1002 | 完整代码

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

int main() {
    int a, b, c;
    cin >> a >> b >> c;          // 按顺序读入三个整数
    cout << b << '\n';           // 只输出第二个
    return 0;
}
```

自测

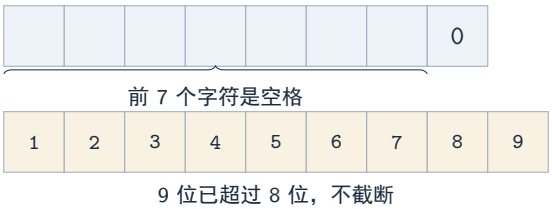
1 2 3 输出 2；-5 0 5 输出 0。

练习 1003 ｜对齐输出：宽度的含义

题目与入口

三个整数各占至少 8 个字符宽度，右对齐；字段紧挨排列。

学校 OJ：P5003 ｜对齐输出



- 0 长度是 1，前面补 7 个空格；-1 长度是 2，补 6 个。
- 三个字段之间不要再手动插入空格；补空格已经决定位置。

练习 1003 | setw 的使用

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

int main() {
    int a, b, c;
    cin >> a >> b >> c;          // 读入三个整数
    cout << setw(8) << a << setw(8) << b << setw(8) << c << '\n';
    return 0;
}
```

- setw(8) 只作用于下一个输出项，所以要写三次。
- 默认右对齐；若题目要求左对齐，使用 left（了解）。

自测：输入 0 -1 123456789，数一数每段前面的空格数。

练习 1004 ｜字符三角形：先画输出草稿

题目与入口

输入字符 c，输出三行等腰三角形。

学校 OJ：P5004 ｜字符三角形

空	空	c	c	
空	c	c	c	
c	c	c	c	c

行	行首空格	字符数
1	2	1
2	1	3
3	0	5

本页要点

还没学循环；只有三行，直接写三条输出语句。

练习 1004 | 代码与空格检查

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

int main() {
    char c;
    cin >> c;                              // 读入一个字符
    cout << "  " << c << '\n';             // 2 个空格 + 1 个字符
    cout << " " << c << c << c << '\n';    // 1 个空格 + 3 个字符
    cout << c << c << c << c << c << '\n'; // 0 个空格 + 5 个字符
    return 0;
}
```

输入 $*$ 时，三行前导空格分别是 $2, 1, 0$ 个。

检查

第三行没有前导空格；每行末尾都要 `'\n'`。

练习 1005 | 地球人口承载力：变量别弄反

题目与入口

输入 x, a, y, b。资源可供 x 亿人生活 a 年，也可供 y 亿人生活 b 年。求长期可持续人口 z。

学校 OJ：P5005 | 地球人口承载力估计

- R：初始资源，单位是“亿人 $\times$ 年”。
- z：每年新增长的资源可支持的人口数。
- 两种情形给出 $R + az = ax$ 和 $R + bz = by$。

先对齐变量

输入的四个数顺序是 x, a, y, b，不要按“先 x 后 y”的习惯读错。

练习 1005 | 公式推导和样例

$$
\begin{aligned}
R + az &= ax, \\
R + bz &= by, \\
(b - a)z &= by - ax, \\
z &= \frac{by - ax}{b - a}.
\end{aligned}
$$

样例代入

$x = 110, a = 90, y = 90, b = 210$，得到 $z = \dfrac{210 \times 90 - 90 \times 110}{210 - 90} = 75$，应打印 `75.00`。

练习 1005 | 完整代码

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

int main() {
    long long x, a, y, b;
    cin >> x >> a >> y >> b;            // 注意输入顺序
    double z = 1.0 * (b * y - a * x) / (b - a);
    cout << fixed << setprecision(2) << z << '\n';
    return 0;
}
```

`1.0 *` 产生小数；`long long` 承接乘法中间结果。

检查

先算分子、再除以分母；括号不能漏，否则整数除法会先截断。

四道练习的常见错误对照

题号	常见错误	正确做法
1002	输出了 a 或 c	只输出 `b`
1003	字段之间手动加空格	三个 `setw(8)` 紧挨
1004	前导空格数写反	第 1 行 2 个、第 2 行 1 个、第 3 行 0 个
1005	变量顺序读错、整数除法	按 x, a, y, b 读，分子乘 `1.0`

- 每题提交前，先用样例输出逐字符比对。
- 样例通过后，再测 0、负数、较大数等边界。

本页要点

评测结果里 `WA` 看答案，`PE` 看空格与换行，`CE` 看编译信息。

常见错误：看懂评测结果再修改

- 编译错误（**CE**）：检查中文标点、分号、花括号、头文件、变量名大小写。
- 答案错误（**WA**）：先手算样例；检查公式、输出空格、小数位数。
- 格式错误（**PE**）：用可见空格检查对齐题，数清三角形每行字符数。
- 运行错误（**RE**）：检查数组越界、除以 0（本章较少出现）。
- 只过样例：再测 0、负数、较大整数等允许的边界输入。

本页要点

提交后先看具体结果，再回到输入、公式、输出三个环节定位。

提交前检查清单

1. 头文件是否齐全？cin/cout 用 <iostream>，格式控制用 <iomanip>。
2. 变量是否都初始化？类型是否足够大？
3. 每条语句是否以英文分号结束？花括号是否成对？
4. 输出是否有提示语、多余空格、多余空行？
5. 小数位数、右对齐宽度是否与题目一致？
6. 是否在本机用样例和至少一组边界数据测试过？

最后一步

把代码从头到尾读一遍，重点看变量名有没有写错、公式有没有漏括号。

本章全部练习：按顺序完成

1. 学校 OJ：P5002 | 输出第二个整数：练习变量与多项输入。
2. 学校 OJ：P5003 | 对齐输出：练习固定宽度、右对齐。
3. 学校 OJ：P5004 | 字符三角形：练习字符与多行输出。
4. 学校 OJ：P5005 | 地球人口承载力估计：练习列方程与两位小数。

完成标准

每题能说明输入是什么、核心式子是什么、为什么这样输出；独立写出完整程序并通过学校 OJ。

本页要点

四题全部通过后，再进入第二章“顺序结构程序设计”。

课后安排与下一章预告

- 复习：把本章第一个完整程序默写一遍，不看课件。
- 练习：四道 OJ 题至少各提交一次，记录每次的评测结果。
- 预习：下一章学习 `long long`、`char`、`bool`、`string` 的取值范围与常用运算。
- 习惯：每写完一个程序，先编译、再运行、最后对照样例。

本章一句话

程序 = 输入 + 处理 + 输出；先把这三件事说清楚，再写代码。

谢谢！