第一章 C++ 程序设计基础

本章内容

- 1.1 程序设计的目标和流程
- 1.2 简单数学运算
- 1.3 变量与常量

1.1 程序设计的目标和流程

核心知识

程序设计的目标，是把现实问题中的处理步骤转化为计算机能够准确执行的指令。一个程序通常要完成三件事：

1. **接收数据**：获得题目给出的输入。
2. **处理数据**：按照公式或规则计算。
3. **给出结果**：严格按要求输出答案。

C++ 属于高级语言。我们编写的源代码不能直接由计算机执行，必须先经过编译。完整流程可以概括为：

```
分析问题 → 设计算法 → 编写源代码 → 编译与链接 → 运行程序 → 测试结果
```

“编译成功”只表示代码符合语法规则并成功生成程序，不表示计算结果一定正确。

关键语法与规则

最小的 C++ 程序结构：

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

int main() {
    cout << "Hello, World!" << endl;
    return 0;
}
```

各部分作用：

代码	作用
`#include <iostream>`	引入标准输入输出流库，使程序可以使用 `cin`、`cout`
`using namespace std;`	允许直接使用标准命名空间 `std` 中的工具
`int main()`	主函数，也是程序的执行入口
`{ ... }`	标出一个代码块的范围，左右花括号必须配对
`cout << ...;`	向控制台输出数据

代码	作用
`return 0;`	结束 `main`，向系统报告程序正常结束

书写代码时还要注意：

- C++ 区分大小写，`cout` 与 `Cout` 不是同一个名称。
- 代码中的标点应使用英文半角符号。
- 大多数语句以英文分号 `;` 结束。
- 一个可独立运行的程序必须有且只有一个 `main` 函数。
- `#include` 是预处理指令，末尾不写分号。

注释供程序员阅读，编译器会忽略注释内容：

```
// 单行注释：从 // 开始到本行末尾

/*
   块注释：可以跨越多行
   适合说明一段代码的整体功能
*/
```

例题

题目： 输出两行文字，第一行输出 `C++ Review`，第二行输出 `Ready!`。

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

int main() {
    cout << "C++ Review" << endl;
    cout << "Ready!" << endl;
    return 0;
}
```

说明： 程序从 `main` 开始执行。`cout` 负责输出，`endl` 负责换行，每条输出语句都以分号结束。

常见错误

- 把 `cout` 使用的 `<<` 写成 `>>`。
- 使用中文分号 `；`、中文引号 `“ ”` 或中文括号。
- 漏写分号，或遗漏 `}`，使括号无法配对。
- 变量名或工具名拼写错误。
- 在 OJ 题目中擅自输出题目没有要求的内容。
- 编译器标出的行不一定是真正写错的位置。遇到括号、分号问题时，应同时向前检查。

1.2 简单数学运算

核心知识

C++ 的基本算术运算符包括：

运算符	作用	示例	结果
`+`	加法	`5 + 2`	`7`
`-`	减法	`5 - 2`	`3`
`*`	乘法	`5 * 2`	`10`
`/`	除法	`7 / 2`	`3`
`%`	求余	`7 % 2`	`1`

两个整数相除时执行**整数除法**，小数部分直接舍去，不进行四舍五入。只要参与除法的一方是浮点数，才会得到浮点结果。

关键语法与规则

```
cout << 7 / 2;        // 3：整数除法
cout << 7.0 / 2;      // 3.5：浮点除法
cout << 14 % 4;       // 2：求余
cout << (2 + 3) * 4; // 20：括号优先
```

常见优先级从高到低：

1. 括号 `()`
2. 乘、除、取余 `* / %`
3. 加、减 `+ -`

`cout` 中双引号会影响输出含义：

```
cout << "2 + 4"; // 输出文本：2 + 4
cout << 2 + 4;   // 输出计算结果：6
```

`%` 和整数除法经常配合使用：

```
int n = 479;
int ones = n % 10; // 9：提取最右边一位
n = n / 10;        // 47：去掉最右边一位
```

可以把 `% 10` 记作“个位提取器”，把 `/ 10` 记作“个位削减器”。

部分数学函数需要引入头文件 `<cmath>`：

```cpp
#include <cmath>

double x = sqrt(81);   // 计算根号 81，结果为 9
double y = pow(2, 5);  // 计算 2 的 5 次方，结果为 32
```

例题

题目： 14 个苹果平均分给 4 名同学，输出每人分到的数量和剩余数量，中间用一个空格分隔。

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

int main() {
    cout << 14 / 4 << " " << 14 % 4;
    return 0;
}
```

输出：

```
3 2
```

说明： `14 / 4` 得到整数商 3，`14 % 4` 得到余数 2。空格属于文本，因此写成 `" "`。

常见错误

- 误以为 `7 / 2` 会得到 `3.5`。需要写成 `7.0 / 2` 或 `7 / 2.0`。
- 对 `double` 使用 `%`。初学阶段把 `%` 理解为整数求余。
- 把 `2 ^ 3` 当作 2 的 3 次方。C++ 中 `^` 不是乘方运算。
- 忘记括号，导致计算顺序与题意不同。
- 把算式放入双引号，程序只会原样输出算式文本。
- 输出多个答案时漏掉题目要求的空格或换行。

1.3 变量与常量

核心知识

变量用于保存输入数据、计算结果和程序运行过程中的中间状态。变量可以理解为内存中带有名称的数据容器。

使用变量通常包含三个动作：

1. **定义**：说明变量的类型和名称。
2. **初始化或赋值**：把数据保存到变量中。
3. **读取或更新**：使用变量中的值，或用新值覆盖旧值。

`=` 是赋值运算符。程序会先计算等号右边，再把结果保存到左边变量中。它与数学中的“相等”含义不同。

常量用于保存程序运行期间不允许改变的数据。常量使用 `const` 定义，并且必须在定义时初始化。

关键语法与规则

变量定义、初始化与更新：

```
int a = 10;           // 定义并初始化
int b, c;             // 同时定义两个 int 变量
b = 3;                // 给 b 赋值
a = a + 5;            // 先计算右边，再把 15 赋给 a
a += 2;               // 等价于 a = a + 2
```

常见的复合赋值(将运算和赋值合并，一种简便写法)：

写法	等价写法
`a += b;`	`a = a + b;`
`a -= b;`	`a = a - b;`
`a *= b;`	`a = a * b;`
`a /= b;`	`a = a / b;`
`a %= b;`	`a = a % b;`

变量命名规则：

- 只能由字母、数字和下划线组成。
- 不能以数字开头。
- 不能使用 `int`、`return`、`if` 等 C++ 关键字。
- 同一作用域内不能重复定义同名变量。
- 建议变量名见名知意，例如 `score`、`totalMinutes`、`radius`。

常量定义：

```
const double PI = 3.14159;
// PI = 3.14; // 错误：常量不能再次赋值
```

典型例题

题目： 输入圆的半径，输出圆的周长和面积。

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

int main() {
    const double PI = 3.14159;
    double r;
    cin >> r;

    double length = 2 * PI * r;
    double area = PI * r * r;
```

```
    cout << length << " " << area;
    return 0;
}
```

说明： 半径和计算结果可能含小数，因此使用 `double`。圆周率在程序运行期间不应改变，因此定义为常量。

变量覆盖示例：猴子吃桃倒推。

```
int peach = 1;                  // 第四天剩 1 个
peach = (peach + 1) * 2;        // 倒推第三天
peach = (peach + 1) * 2;        // 倒推第二天
peach = (peach + 1) * 2;        // 倒推第一天
cout << peach;                  // 22
```

每次赋值都先使用 `peach` 的旧值完成右侧计算，再用结果覆盖旧值。

常见错误

- 变量没有定义就使用。
- 变量未初始化就参与计算，可能得到不可预测的值。
- 把 `=` 当作判断相等。赋值写 `a = 5;`，判断相等写 `a == 5`。
- 写 `a + 1;` 却以为 `a` 会自动变化。更新变量应写 `a = a + 1;`、`a += 1;` 或 `a++;`。
- 常量没有初始化，或在定义后再次赋值。
- 变量名以数字开头、包含中文或使用了关键字。

本章练习

一、基础知识

1. C++ 程序从哪个地方开始执行？

2. `#include <iostream>` 的作用是什么？它的末尾是否需要分号？

3. 单行注释和块注释分别怎样书写？

4. 分别写出加、减、乘、除、求余对应的运算符。

5. 计算下列表达式的结果：
    - `13 / 5`
    - `13 % 5`
    - `13.0 / 5`
    - `(2 + 3) * 4`

6. 判断下列变量名是否合法，并说明原因：
   - `score1`
   - `1score`
   - `total_score`
   - `double`
   - `studentAge`
7. `=` 在 C++ 中表示什么？`a = a + 1;` 应怎样理解？

8. 为什么 `const double PI = 3.14159;` 必须在定义时给出初始值？

二、代码阅读

1. 写出程序的输出结果：

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

int main() {
    cout << "3 + 4" << " " << 3 + 4;
    return 0;
}
```

2. 写出程序的输出结果：

```cpp
int a = 5;
a = a + 3;
a *= 2;
cout << a;
```

3. 找出下面代码中的错误并改正：

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

int Main {
    cout >> "Hello";
    return 0
}
```

4. 写出 `ones` 和 `n` 的最终值：

```
int n = 628;
int ones = n % 10;
n = n / 10;
```

三、程序设计

1. **两数运算**

   输入两个整数 `a` 和 `b`，依次输出它们的和、差、积，三个结果之间用一个空格分隔。

   - 样例输入：`8 3`
   - 样例输出：`11 5 24`

2. **平均分苹果**

   输入苹果总数 `m` 和学生人数 `n`，输出每人分到的苹果数以及剩余苹果数，中间用一个空格分隔。题目保证 `n > 0`。

   - 样例输入：`17 5`
   - 样例输出：`3 2`

3. **圆的周长与面积**

   输入圆的半径 `r`，使用常量 `PI = 3.14159`，输出圆的周长和面积，中间用一个空格分隔。