第1章　顺序结构

本章是进入 C++ 魔法世界的第一站，叫作顺序结构。与我们平时讲故事、写作文一样，要一步一步来，程序也是从上往下一行一行执行的。

先学习搭建程序的“小城堡”（程序骨架），并亲手写出第一句代码“Hello，C++！”，向计算机世界问好。接着认识几个重要的“小帮手”：专门装整数的“整数盒子”（int）、能处理小数的“小数盒子”（double），还有能记住字母和符号的“字符盒子”（char）。此外，我们还会发现字符背后藏着的小秘密——ASCII 码。

学会了存数据，就要学习怎样和计算机对话啦！例如，学习用 cin 把数据告诉计算机，用 cout 让计算机把结果告诉我们。不仅如此，我们还要学怎么做数学题，加、减、乘、除都难不倒计算机。特别要注意除法和取余数（%）的用法，这可是解决很多数学趣题（比如拆分数字、算周期）的超级法宝。通过学习本章内容，你就能指挥计算机帮你做算术题、解决生活中的小问题啦！

本章任务

（1）会写代码：能背下程序的“万能骨架”，写出能运行的 C++ 小程序。
（2）认识“变量盒子”：知道怎么定义 int（整数）变量，明白变量就像个小盒子，专门用来存储数据。
（3）学会和计算机对话：熟练使用 cin（输入）向计算机提供数据，用 cout（输出）让计算机显示答案。
（4）变成“算术小能手”：会用计算机做加（+）、减（-）、乘（*）法运算，解决简单的数学题。
（5）搞定“小数点”：学会用 double 存小数，还能指挥计算机保留几位小数（比如保留两位小数）。
（6）分清两种“除法”：明白在计算机里“10 除以 3”和“10.0 除以 3”是不一样的，学会怎么算除法。
（7）掌握“取余数”法宝：学会用 % 求余数，能用它来把一个多位数的个位、十位拆解出来。
（8）玩转“字符密码”：认识 char 类型，知道字母和数字在计算机里都有一个神奇的编号（ASCII 码）。
（9）让输出更漂亮：学会用 printf 或者 setw 等工具，把输出的数字排排队，打印得整整齐齐。

1.1　初识 C++ 与第一程序

一、学习目标

（1）掌握 C++ 程序的基本框架，并能默写“万能骨架”。
（2）理解 cout 输出语句的语法，并能使用它在屏幕上显示文字和数字。
（3）能够编写并运行自己的第一个 C++ 程序：“Hello, C++!”

二、知识讲解

1. C++ 程序的基本框架

每个 C++ 程序都必须包含一个固定的“骨架”，它是程序能够运行的最低要求。

就像玩乐高，无论想拼汽车还是城堡，都得先有一块基础底板。这个“骨架”就是 C++ 程序的“底板”。

每个程序开发人员都必须能够默写标准的“万能骨架”代码。

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

int main() {
    // 你的所有 " 魔法指令 " 都写在这里
    return 0;
}
```

2. 框架解析

1）#include <iostream> （导入工具箱）

作用：告诉编译器“我需要使用输入（cin）和输出（cout）功能”。

iostream 是一个装满“输入输出工具”的工具箱，#include 表示把这个工具箱搬到你的工作台上。

注意：必须写在程序开头，行尾没有分号。

2）using namespace std;（命名空间声明）

std 的含义：standard（标准）的缩写，代表 C++ 标准库的“品牌名”。

比喻：就像在文具店说“我要买晨光的笔”，说一次“晨光”后，后面就可以直接说“笔、橡皮、尺子”，不用每次都强调品牌。

注意：写在 #include 之后、main 函数之前，行尾有分号。

3）int main()（程序主入口）

作用：程序的唯一入口。所有 C++ 程序都从 main 函数的第一行开始执行。一个程序中有且仅有一个 main 函数。

比喻：如果程序是一本书，main 就是这本书的第一页。

注意：main 必须小写，后面的 () 不能省略。

4）{ }（花括号，即代码围墙）

作用：标记 main 函数的开始和结束。所有要执行的代码都必须写在这对花括号内部。

比喻：这对花括号圈出了 main 函数的地盘。

注意：必须成对出现，缺少任意一个都会导致编译失败。

5）// 注释内容（单行注释）

作用：为代码添加说明和解释，方便自己和他人理解代码的含义。

格式：双斜线 // 后面的所有内容都是注释，只对当前这一行有效。

使用建议：在复杂的代码旁边添加注释，说明“这段代码在做什么”。

6）return 0;（任务完成报告）

作用：告诉操作系统“程序已正常运行结束”。

比喻：就像一个人在完成任务后，向指挥官报告“任务顺利完成”。

注意：通常是 main 函数的最后一条语句，行尾有分号。

3. 什么是输出（cout）语句

cout 是 C++ 中的标准输出对象，专门用来在屏幕上显示信息，是 character output（字符输出）的缩写。

比喻：把 cout 想象成一个“信息广播站”或“屏幕打印机”——把想要展示的内容通过“传送带”送给它，它就会立刻在屏幕上显示出来。这个“传送带”就是 << 符号。

4. 语法格式

```
cout << 你想输出的内容；
```

说明：

（1）cout：代表“屏幕”这个输出设备。

（2）<<：称为“流插入运算符”。

比喻：它像一个传送带，把右边的“货物”（数据）传送到左边的 cout（屏幕）。

（3）; 是语句结束的标志。

5. 具体用法与注意事项

（1）输出文字（字符串）：要输出的文字必须用英文双引号包裹，双引号本身不会被输出。

```
cout << "Hello, C++!"; // 正确
//cout << Hello, C++;  // 错误：未用双引号包裹
```

（2）输出数字：输出整数、小数等数字时，直接写，不需要加双引号。

```
cout << 2024;
cout << 3.14;
```

（3）连续输出：可以用多个 << 连接，一次输出不同类型的内容。

```
cout << "CSP-J " << 2024 << " fighting!"; // 输出：CSP-J 2024 fighting!
```

（4）换行输出：使用 endl（end line 的缩写）可以让光标移动到下一行的开头。

```
cout << "第一行" << endl;
cout << "第二行";
```

三、案例演示

1. 我的第一个程序

【题目描述】编写一个 C++ 程序，在屏幕上输出“Hello, C++!”。

【输入描述】无输入。

【输出描述】输出一行文本“Hello, C++!”。

【输入样例】

```
（无输入）
```

【输出样例】

```
Hello, C++!
```

2. 输出表达式的值

【题目描述】输出表达式 1+2+3+4 的结果。

【输入描述】无输入。

【输出描述】如输出样例所示。

【输入样例】

```
无输入
```

【输出样例】

```
1+2+3+4=10
```

四、AI 辅助学习

1. 代码检查与调试

```
请帮我检查这段 C++ 代码是否有语法错误，并解释每个错误的原因：[ 粘贴你的代码 ]
```

2. 当你想理解一段代码时，让 AI 为你“逐行精讲”

```
请像一位信息学竞赛老师一样，为我逐行解释下面这段 C++ 代码的每一部分都是什么意思，特别是 `#include` 和 `int main()` 的作用。
#include <bits/stdc++.h>
using namespace std;
int main() {
    cout << "Hello, C++!";
    return 0;
}
```

五、活学活用

（1）计算矩形的面积与周长。

【题目描述】长为 8，宽为 5，计算并输出矩形的面积和周长。

【输入描述】无输入。

【输出描述】第一行，矩形的面积是：××。第二行，矩形的周长是：××。

【输入样例】

```
无输入
```

【输出样例】

```
矩形的面积是：40
矩形的周长是：26
```

（2）输出欢迎信息。

【题目描述】编写一个 C++ 程序，在屏幕上输出欢迎信息“Welcome to GESP!”。

【输入描述】无输入。

【输出描述】输出一行文本“Welcome to GESP!”。

【输入样例】

```
无输入
```

【输出样例】

```
Welcome to GESP!
```

（3）矩形字母表。

【题目描述】绘制一个 10×18 的图形，如输出样例所示。

【输入描述】无输入。

【输出描述】输出一个 10×18 的图形。

【输入样例】

```
无输入
```

【输出样例】

```
ABCDEFGHIJKLMNOPQR
BABCDEFGHIJKLMNOPQ
CBABCDEFGHIJKLMNOP
DCBABCDEFGHIJKLMNO
EDCBABCDEFGHIJKLMN
FEDCBABCDEFGHIJKLM
GFEDCBABCDEFGHIJKL
HGFEDCBABCDEFGHIJK
IHGFEDCBABCDEFGHIJ
JIHGFEDCBABCDEFGHI
```

（4）关于 C++ 程序基本框架，下列说法错误的是（　）。

A. 一个 C++ 程序必须包含一个 main 函数

B. #include <iostream> 这行代码的末尾需要加上分号

C. return 0; 表示程序正常结束

D. C++ 语言对大小写敏感，main 和 Main 会被识别为不同的标识符

（5）分析以下代码的输出结果。

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

int main() {
    cout << "GESP" << 2024;
```

```
    cout << endl;
    cout << "CSP-J/S";
    return 0;
}
```

A.

```
GESP2024
CSP-J/S
```

B.

```
GESP 2024
CSP-J/S
```

C.

```
GESP2024CSP-J/S
```

D.

```
GESP
2024
CSP-J/S
```

（6）以下（　）是 C++ 程序必须包含的部分。

A. #include <iostream>　　B. using namespace std;

C. int main()　　D. cout <<“Hello”;

（7）以下代码的输出结果是（　）。

```
int x = 10, y = 5;
cout << "x + y = " << x + y << endl;
```

A. x + y = 10 + 5　　B. x + y = 15

C. x + y = xy　　D. 编译错误

（8）下列代码中，会产生编译错误的是（　）。

A. cout <<“Hello” << endl;　　B. cout << 123 << endl;

C. cout <<“Age: ” << 15 << endl;　　D. cout >>“Hello” << endl;

（9）阅读以下代码，输出结果是（　）。

```
#include <iostream>
using namespace std;
int main() {
    cout << "A" << "B" << "C" << endl;
    return 0;
}
```

A. A B C　　B. ABC

C.“A”“B”“C”　　D. 编译错误

（10）阅读以下代码，输出结果是（　）。

```
#include <iostream>
using namespace std;
int main() {
    int x = 10;
    cout << "x = " << x << ", x + 5 = " << x + 5 << endl;
    return 0;
}
```

A. x = 10, x + 5 = 15　　B. x = x, x + 5 = x + 5

C. x = 10, x + 5 = 10 + 5　　D. 编译错误

1.2　整 型 变 量

一、学习目标

（1）了解变量的概念，掌握整型变量的声明、初始化和赋值方法。

（2）掌握整型变量在程序中的使用方式，能够进行基本的数据存储和运算。

（3）熟练运用变量命名规则，能够为变量起规范且有意义的名字。

二、知识讲解

1. 变量和整型变量

变量：程序中用于存储数据的一块带名字的内存空间。

计算机的内存是一座“摩天大楼”。定义一个变量，就相当于向大楼管理员申请了一间带门牌号的房间。变量名就是这个房间的门牌号（例如 score），而房间里存放的东西就是变量的值（例如 100）。通过门牌号，我们可以随时找到这个房间，查看或更换里面的东西。

整型变量：一种专门用来存放整数的变量。在 C++ 中，我们使用关键字 int 来定义它。如果说变量是储物柜，那么 int 类型的变量就是一个特制的、只能放整数“积木”的柜子，不能把小数（如 3.14）或者文字（如 “hello”）放进去。

2. 整型变量的声明、初始化与赋值

标准语法格式如下。

```
// 格式 1：先声明
int 变量名 ;

// 格式 2：声明时初始化
```

```
int 变量名 = 初始值；

// 格式 3：先声明，后赋值
int 变量名；
变量名 = 值；
```

说明：

（1）int：关键字，告诉编译器这是一个整型变量。

（2）变量名：为变量起的名字，命名时必须遵循一定的规则。

（3）=：赋值运算符，将右边的值存储到左边的变量中。

（4）初始值或值：想要存储在变量中的整数。

（5）; 为语句结束符，不可省略。

3. 具体用法与注意事项

（1）变量的声明（申请房间）。

```
int age;          // 声明一个名为 age 的整型变量
int score;        // 声明一个名为 score 的整型变量
int a, b, c;      // 同时声明 3 个整型变量（逗号分隔）
```

注意：声明后的变量如果没有初始化，里面的值是不确定的（可能是任意数字），就像一个新房间里可能遗留着上一位租客的杂物。

（2）变量的初始化（申请房间并放入物品）。

```
int age = 15;               // 声明 age 并立即赋值为 15
int score = 100;            // 声明 score 并立即赋值为 100
int a = 10, b = 20;         // 同时声明并初始化多个变量
int x = 5, y, z = 10;       // 混合声明：x 初始化为 5，y 未初始化，z 初始化为 10
```

推荐做法：声明变量时尽量同时初始化，避免使用不确定的值。

（3）变量的赋值（更换房间里的物品）。

```
int number;       // 先声明
number = 42;      // 后赋值
number = 100;     // 重新赋值，原来的值 42 被覆盖，现在是 100
```

关键理解：赋值会覆盖原来的值，旧值会消失。

（4）变量的使用（使用房间里的物品）。

```
cout << age;                        // 输出变量的值
int sum = age + brother_age;        // 在表达式中使用变量进行运算
int double_score = score * 2;       // 使用变量参与计算
```

（5）示例。

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int age = 15;                          // 声明并初始化：我的年龄
```

```
    int brother_age = 12;               // 声明并初始化：弟弟的年龄
    int sum = age + brother_age;        // 计算年龄总和

    cout << "我的年龄：" << age << endl;
    cout << "弟弟的年龄：" << brother_age << endl;
    cout << "年龄总和：" << sum << endl;

    return 0;
}
```

输出结果如下。

```
我的年龄：15
弟弟的年龄：12
年龄总和：27
```

4. 变量的命名规则

（1）字符限制：只能包含字母（a ～ z, A ～ Z）、数字（0 ～ 9）和下画线（_）。

```
int age;            // √ 合法
int student_count; // √ 合法
int score2;         // √ 合法
int my-age;         // × 非法：不能使用连字符 -
int student count; // × 非法：不能有空格
```

（2）开头限制：必须以字母或下画线开头，不能以数字开头。

```
int age;            // √ 合法
int _temp;          // √ 合法（但不建议，容易与系统变量混淆）
int 2score;         // × 非法：不能以数字开头
```

（3）关键字限制：不能使用 C++ 的保留关键字。

```
int number;         // √ 合法
int int;            // × 非法：int 是关键字
int cout;           // × 非法：cout 是标准库名称
int return;         // × 非法：return 是关键字
```

常见关键字：int、if、else、for、while、return、break、continue 等。

（4）大小写敏感：C++ 区分大小写，Age 和 age 是两个不同的变量。

```
int age = 15;
int Age = 20;       // 这是另一个变量，与 age 不同
cout << age;        // 输出 15
cout << Age;        // 输出 20
```

5. 变量的推荐规范（良好的编程习惯）

（1）使用有意义的名字：变量名应能表达其存储内容的含义。

```
int s = 100;                // × 不推荐：s 意思不明确
int score = 100;            // √ 推荐：一看就知道是分数
```

```
int x = 30;                 // × 不推荐：x 含义不明
int student_count = 30;     // √ 推荐：表示学生数量
```

（2）使用下画线命名法或驼峰命名法。

```
// 下画线命名法（推荐用于 C++）
int student_count;
int max_score;

// 小驼峰命名法（首字母小写）
int studentCount;
int maxScore;

// 大驼峰命名法（首字母大写，通常用于类名）
int StudentCount;   // 变量一般不用这种
```

（3）避免使用单个字母（除非是常见约定）。

```
int i, j, k;        // √ 可接受：循环计数器的惯例
int n, m;           // √ 可接受：表示数量的惯例
int a, b, c;        // × 不推荐：含义不明确
```

三、案例演示

1. 苹果篮子

【题目描述】小明的苹果篮子里现在有 8 个苹果。请你编写一个程序，计算当他再往篮子里放入 12 个苹果后，篮子里总共有多少个苹果，并输出这个总数。

【输入描述】无输入。

【输出描述】程序应输出一个整数，代表篮子里的苹果总数。

【输入样例】

```
无输入
```

【输出样例】

```
20
```

2. 变量交换

【题目描述】假设有两个杯子，A 杯子里装的是牛奶，B 杯子里装的是果汁。现在你想把杯子里的东西交换一下，在 A 杯子里装果汁，在 B 杯子里装牛奶。你该怎么做？ 在程序中，有两个整型变量 a 和 b，初始值分别为 100 和 200。请你编写一段代码，交换这两个变量的值，然后分两行输出它们交换后的值。

【输入描述】无输入。

【输出描述】第一行输出变量 a 的新值，第二行输出变量 b 的新值。

【输入样例】

```
无输入
```