C++ 输入（`cin`）讲义

一、 程序只能算固定数据吗？

我们之前写的代码，所有的数字都是直接写在程序里的，比如 `int a = 5; int b = 3;`。这种写法把**数据写死**了。
大家思考一下，如果用户明天想算 `100 + 200` 怎么办？难道每次都要让程序员打开代码，改完数字再重新算一次吗？这显然是不现实的！
一个真正有价值的程序，应该是**智能的**——它能够随时接收用户提供的新数据，并自动计算出结果。今天，我们就要学习如何让程序接收用户提供的数据。

二、 认识 `cin`：控制台的数据接收员

为了让程序接收用户的键盘输入，C++ 提供了一个强大的工具：`cin`。

- **含义**：Console Input（控制台输入）。
- **作用**：**让用户在黑框框里用键盘输入数据**，然后程序会自动把这个数据存入我们提前准备好的变量中。
- `cin` **使用 >> 运算符跟上准备好的变量来接收用户输入。**

1. `cin` 的运作过程

我们来看看 `cin` 是如何与我们之前学习的变量配合工作的：

```
int num;
cin >> num;
cout << num;
```

1. `int num;` → 程序向内存申请了一个整数类型的空柜子，并在外面贴上标签 `num`。
2. `cin >> num;` → 程序暂停运行，等待用户操作。用户在键盘上敲下数字 `7` 并按下回车。`>>`**（提取符）**就像传送带一样，把键盘上的 `7` 抓过来，放进了名为 `num` 的抽屉里。
3. `cout << num;` → 获取名为 `num` 抽屉里的数据，并将其展示在屏幕上（输出 7）。

2. 语法与规则

语法格式：`cin >> 变量名;`

如果你想一次性让用户输入多个数据，可以把传送带连起来用：

```
int a, b;
cin >> a >> b; // 依次接收用户输入的两个整数
```

> **输入规则：**
> 当使用 `cin >> a >> b;` 时，用户在黑框框里输入数据，**必须使用“空格”或者“回车”来隔开两个数字**（比如输入 `5 10`）。`cin` 它只要一遇到空格或回车，就知道第一个数据输入结束了，准备把下一个数据放进第二个抽屉里。

三、 解决题目中要求有输入的题目

现在我们拥有了所有的基础工具（变量、加减乘除、输入、输出）。接下来，我们要做一道完整的带有输入程序题。

题目描述：请你实现输入两个整数，输出这两个整数的差。
(样例输入：2 3 → 样例输出：-1；样例输入：10 2 → 样例输出：8)

分析思路：

1. **提取有效信息**：题目需要什么？（**输入两个数** → 相减 → 输出结果）。
2. **整理思路**：

- 准备两个空抽屉装输入的数。
- 用 `cin` 把用户敲的字放进抽屉。
- 拿出来相减，结果放进第三个抽屉。
- 用 `cout` 展示第三个抽屉。

3. **C++ 代码**：

```
int num1, num2;            // 准备抽屉
cin >> num1 >> num2;       // 接收输入
int diff = num1 - num2;    // 计算并存入新抽屉
cout << diff;              // 输出结果
```

4. **编译运行**：按下 F11。如果有报错，不要慌，检查是不是漏了分号，或者 `>>` 和 `<<` 的方向写反了。
5. **输入测试数据，验证结果**：在弹出的黑框里输入测试用例 `2 3`，看看结果是不是 `-1`。
6. **修正错误**：如果输出不对（比如输出了 5），说明语法没问题，但是**逻辑错了**。回去检查代码，是不是把减号写成加号了？修改后重新测试，直到完全正确。

四、 流程图

如果我们全用一段段的文字来写思路，一眼望过去密密麻麻，很不直观。

在软件工程中，**算法**（解决问题的方法和步骤）才是核心，而代码只是这些步骤的最终翻译。为了更好地设计和理解算法，工程师们发明了一种世界通用的图形化语言——**流程图 。

流程图使用预定义的符号：

- **圆角矩形代表开始/结束**
- **平行四边形代表输入/输出**
- **长方形代表计算处理**
- **菱形代表判断**

图框	名称	功能
圆角矩形	起止框	表示一个算法的开始和结束
平行四边形	输入输出框	表示一个算法的输入和输出信息
矩形	处理框	赋值，执行计算语句，结果的传送
菱形	判断框	表示判断某一个条件是否成立
→ ↓	流程线	表示执行步骤的路径流程进行的方向

把我们大脑里抽象的逻辑，变成了一张图。以后面对复杂的题目，画出流程图，思路就非常清晰，例如如何把大象塞进冰箱的流程图为：

