C++ 格式化输入与输出 讲义

一、 引言

问题引入：数字反转

【书本P21页 例2-7】我们来看一个问题：

> 输入一个三位浮点数，例如 `123.4`，要求把这个数字彻底翻转过来，变成 `4.321` 并输出。

乍一看，这个问题似乎很难下手。我们熟悉的 `cin` 只能整个地读取这个数字，`cout` 也只能整个地输出。我们如何能把一一`1`、`2`、`3`、`.`、`4`一一拆开，再按新的顺序重新组合呢？

二、 解法一：数学运算

在揭晓新工具之前，我们先用已有的知识尝试解决。

思路铺垫

对于 `123.4`，最碍事的就是那个小数点。如果我们能先把它变成整数 `1234`，问题就简化成了“整数拆分”。

核心工具：提取器与削减器

我们回顾一下拆分整数的两个工具：

1. **个位的提取器 (`% 10`)**
    - **作用**：获取一个整数的**最右边一位。**
    - **原理**：任何整数除以10的**余数**，正好就是它的个位数。
    - **示例**：`1234 % 10` 结果是 `4`。
2. **数字的削减器 (`/ 10`)**
    - **作用**：**砍掉**一个整数的**最右边一位。**
    - **原理**：在C++中，`整数 / 整数` 的结果会**直接舍弃小数部分。**
    - **示例**：`1234 / 10` 结果是 `123`。

步骤拆解

1. **第一步（提取）**：使用 **提取器 (`% 10`)** 获得当前数字的个位数。
2. **第二步（削减）**：使用 **削减器 (`/ 10`)** 砍掉个位数，得到一个更小的数，为下一次操作做准备。

具体案例

让我们来手动拆解整数 `479`。

分步演示：

- **初始状态**: 我们有一个整数 `n = 479`。
- **第一轮操作 (处理 `479`)**:
    1. **提取**: `479 % 10` -> 我们得到了数字 `9`。
    2. **削减**: `479 / 10` -> `n` 现在变成了 `47`。

- *我们成功地把 `9` 从 `479` 中分离了出来，并且得到了一个更简单的问题：如何拆解 `47`。*

- **第二轮操作 (处理 `47`)**:
  1. **提取**: `47 % 10` -> 我们得到了数字 `7`。
  2. **削减**: `47 / 10` -> `n` 现在变成了 `4`。
  - *小结：我们又分离出了 `7`，问题变得更简单了：如何拆解 `4`。*

- **第三轮操作 (处理 `4`)**:
  1. **提取**: `4 % 10` -> 我们得到了数字 `4`。
  2. **削减**: `4 / 10` -> `n` 现在变成了 `0`。
  - *小结：我们分离出了最后的 `4`，`n` 变成了 `0`，说明拆解工作已全部完成！*

而对于我们的题目：**输入一个三位浮点数，例如 `123.4`，要求把这个数字彻底翻转过来，变成 `4.321` 并输出。**

我们就也能通过整数拆分的办法来解决了。

1. **预处理**：`double num = 123.4; int n = num * 10;` -> 得到整数 `1234`。
2. **逐位拆分**：
   - `int d = n % 10;` // d = 4
   - `int c = (n / 10) % 10;` // c = 3
   - `int b = (n / 100) % 10;` // b = 2
   - `int a = n / 1000;` // a = 1
3. **重新组合输出**：
   - `cout << d << "." << c << b << a;`

结论：通过整数拆分的方法可以解决本问题，且一定要掌握通过数学运算拆分整数的办法。

常见问题

1. **问：对一个个位数（比如 `8`）使用这两个工具会怎样?**
   - **答**：完全符合规则，不会出问题！
     - **提取器**：`8 % 10` 的结果是 `8`。（8除以10，商0余8，成功提取出了最后一位数字 8）
     - **削减器**：`8 / 10` 的结果是 `0`。（8除以10，整数商是0，成功削减了最后一位数字 8）
2. **问：如果数字里有 `0` 呢? 比如 `205`?**
   - **答**：规则同样适用！
     - **第一轮**: `205 % 10` -> `5`； `205 / 10` -> `20`。
     - **第二轮**: `20 % 10` -> `0`； `20 / 10` -> `2`。
     - **第三轮**: `2 % 10` -> `2`； `2 / 10` -> `0`。

要点回顾

1. **个位的提取器 (`% 10`)**：它的作用就是帮你**看到并取走**一个整数最右边的那一位。
2. **数字的削减器 (`/ 10`)**：它的作用就是**砍掉**一个整数最右边的那一位，让数字变小。
3. 而且“提取”后必须“削减”，这样才能为处理下一位数字做准备。它们总是成对出现。

三、 `printf`

核心概念与语法

`printf` 的全称是 **Print Formatted（格式化打印）**。
它的作用是：**先写好一段带有“空位”的模板句子，然后再把变量按顺序填入这些空位中。**

语法格式：

> `printf("包含格式说明符(占位符)的字符串", 变量1, 变量2, ...);`

`printf` 的使用方法

1. **打印纯文本**

```
printf("Hello, World!\n"); // \n 代表换行
```

2. **打印单个变量**
`printf` 在于用**格式说明符**（以`%`开头的占位符）去指定“空位”的类型，也就是先挖个坑占个座，之后我来告诉你什么数据会放在这里你进行输出就可以了。

```
int age = 30;
// %d 是专门用来放整数的占位符
printf("我的年龄是 %d 岁。\n", age);
```

逻辑解释：

1. `"我的年龄是%d岁。\n"` 是输出的模板字符串。
2. `%d` 是一个占位符。
3. `printf` 会在输出时，自动按照顺序用后面 `age` 变量的值（30）去替换 `%d` 的位置。

3. **混合打印多个变量**

`printf` 也可以极其方便地在一句话中同时输出多个、多种类型的变量，而不需要像 `cin` 或 `cout` 那样反复使用 `<<` 拼接。

```
int age = 15;
double height = 168.5;
char rank = 'S';

// 依次使用 %d, %lf, %c 作为占位符
printf("我今年%d岁，身高是%lf厘米，综合评级：%c\n", age, height, rank);
```

核心规则：

- **一一对应：** 字符串里有 3 个坑（`%d`, `%lf`, `%c`），后面就必须跟着 3 个变量（`age`, `height`, `rank`）。

- **按顺序入座：** `printf` 会严格按照从左到右的顺序，把第一个变量填入第一个占位符，第二个变量填入第二个占位符，以此类推。
- **类型必须匹配：** 第一个是整数类型 `age`，所以必须用 `%d`；第二个是小数类型 `height`，必须用 `%lf/%f`；第三个是字符类型 `rank`，必须用 `%c`。

常用格式说明符（书本P24页，重要！）

占位符的顺序必须与后面变量的顺序一一对应，且数据类型必须兼容！

占位符	对应数据类型	说明
`%d`	`int`	打印整数
`%f`	`float`	打印单精度小数 (默认保留6位小数)
`%lf`	`double`	打印双精度小数 (打印小数一般使用 `%f` 即可)
`%c`	`char`	打印单个字符
`%.mf`	`float/double`	**技巧：** 打印浮点数，并**精确保留m位小数。**例如 `%.2f` 就是保留两位。

`cout` 与 `printf` 写法对比

操作目标	`cout` 写法	`printf` 写法
打印纯文本	`cout << "输出的内容";`	`printf("输出的内容");`
打印单个变量	`cout << num1;`	`printf("%d", num1);`
混合打印多个变量	`cout << num1 << "和" << num2;`	`printf("%d和%d", num1, num2);`

四、 `scanf`

概念与语法

`scanf` 的全称是 **Scan Formatted（格式化扫描/输入）。**
它的逻辑和 `printf` 基本一致，只不过从输出数据 变成了 输入数据：它会根据你提供的格式字符串模板，去键盘的输入中**抓取**对应的数据。

语法格式：

```
scanf("格式控制字符串", &变量1, &变量2, ...);
```

要注意 `scanf` 必须加 `&`（取地址符）

原理：

- `cin` 很智能，你告诉它抽屉的名字（如 `age`），它就能自己找到。
- `scanf` 是一位老员工，你必须把这个抽屉在内存中的**具体物理地址**交给它，它才能把数据放进去。

那么在变量名前加上 `&`，就相当于提取了这个变量的地址，所以这个符号也被称为取地址符！

> **警告：** 如果忘记写 `&`，程序在运行到输入阶段时会直接崩溃！

基础使用示例：

```
int age;
printf("请输入您的年龄：");
scanf("%d", &age); // 从键盘输入获取一个整数，存入 age 所在的内存地址
// 现在 age 中就储存了输入的数据
```

`scanf` 的使用技巧

1. **一次性读取多个数据**

   ```
   int a;
   double b;
   scanf("%d%lf", &a, &b); // 从键盘输入中读取一个整数和一个double类型的数据
   ```

   用户输入时用空格、Tab或回车 (空白字符) 隔开即可，`scanf` 会自动跳过它们。

2. **易错点：不要在格式字符串里乱加标点，除非题目格式明确要求了**
   - **错误写法**：`scanf("%d,%d", &a, &b);`
   - **后果**：那么用户输入时**必须严格**按照格式输入：**[一个整数 逗号 一个整数]** 的格式，例如 `5, 10` 。如果不带逗号输入 `[5 10]` 的话就会读取失败。
   - **正确、安全的写法**：`scanf("%d%d", &a, &b);` 保持格式字符串的干净，只保留占位符。

五、 实战演练

实战1：数字反转

解题思路：我们不把 `123.4` 看作数字，而是看作“3个字符 + 1个小数点 + 1个字符”。利用 `scanf` 的格式匹配能力，将它们分离到不同的`char`变量中！

```
#include <cstdio>

int main() {
    char a, b, c, d;
    // 输入格式为 %c%c%c.%c，匹配 [字符 字符 字符 小数点 字符] 的格式将4个字符分离出来
    scanf("%c%c%c.%c", &a, &b, &c, &d);

    // 按照 d.cba 的倒序格式输出
    printf("%c.%c%c%c\n", d, c, b, a);
    return 0;
}
```

用 `scanf` 和 `printf`问题就会变得非常简单！

实战2：再分肥宅水（书本P23页）

题目要求：t 毫升水均分给 n 名同学。输出每人获得多少毫升（严格保留3位小数）以及一共需要多少个杯子（每人2个）。

```
#include <cstdio>
```

```c
int main() {
    double t;
    int n;

    // 用 %lf 读 double，用 %d 读 int
    scanf("%lf%d", &t, &n);

    // t/n 对应 %.3f (保留三位小数)， n*2 对应 %d，中间用 \n 换行
    printf("%.3f\n%d\n", t / n, n * 2);

    return 0;
}
```

六、 总结

- `cin/cout`：语法简单直观，能自动识别类型，是我们日常编程的常用选择。
- `scanf/printf`：
  - **优点**：执行效率更高；拥有强大的**格式控制能力**，能实现精准的输入匹配和带精度的输出。
  - **缺点**：语法稍显复杂，需要手动匹配占位符和变量类型，并且 `scanf` 必须带 `&`。
- **如何选择?** 在需要精确控制输入输出格式（如本讲的例题），就非常推荐使用 `scanf/printf`。