

03 | 判断与循环：给你的程序加上处理逻辑

2020-01-09 胡光

人人都能学会的编程入门课

[进入课程 >](#)



讲述：胡光

时长 21:04 大小 16.89M



你好，我是胡光，咱们又见面了。不知道上一讲的内容，你自己做练习了么？你是否还觉得 C 语言枯燥无味呢？不管你有没有练习，我都还要啰嗦下，学习编程，就像是学骑自行车，你只看别人怎么骑，你只看自行车的原理，那永远也不可能学会骑自行车，对于你来说，唯一的捷径就是多练习，多思考。在上一讲小试牛刀之后，今天我将带你领略一下算法和逻辑层面的小惊喜。


今日任务

先来看看今天这 10 分钟我们要完成的任务。日期这个概念你肯定不陌生，生日对你我来说都很重要，如果你身边有 2 月 29 号过生日的小伙伴，恐怕最少 4 年，才能为他 / 她办一

次生日宴。今天我们的这个任务，就和日期有关系。如果我给你一个由年月日组成的日期，再给你一个数字 X，你能否准确地让程序输出 X 天后的日期呢？

例如下面这个数据：

```
1 1989 11 20
2 20
3 1989 12 10
```

 复制代码

数据中给出了 1989 年 11 月 20 日这个日期，然后问你 20 天后的日期是多少，你的程序应该输出 1989 年 12 月 10 日。特别需要注意的是，在这个任务中，你需要考虑到闰年中 2 月份的特殊性，闰年的 2 月有 29 天。今天我们就学习，如何用计算机解决这类任务吧。

必知必会，查缺补漏

根据对任务的理解，我们可以分成两步来思考这个问题：

第一步：我们来思考如何求解 1 天后的日期，在求解 1 天后日期的过程中，我们涉及到的核心问题就是判断日子是否跨月，月份是否跨年，即判断；

第二步：是要让计算机重复 X 次 1 天后日期的计算过程，即重复循环做这件事。

要解决这两个难题，我们需要讲讲 C 语言中的一些基础知识，其中包括了程序中用于**逻辑分支判断的分支结构**，以及可以**重复做大量事情的循环结构**。听着这些专业词汇，你可能有点懵，别怕，等我下面讲了它们是什么意思，你就会感觉这些其实很简单。


1. 给代码添加判断能力：“if...else”语法结构

我们一起来读下这句话：如果隔壁商店有酱油，就买酱油，否则就买点儿醋回来。可以看到，这句话用了“如果.....就”的假设关系关联词，“如果”后面接的是假设条件，“就”后面接的是条件成立后的结果，“否则”接的是条件不成立后的结果。

现在我们想把计算机变成我们的小帮手，就必须要有的一种语法，能够表达“如果.....就.....否则.....”的逻辑，这种语法，就是接下来我要介绍给你的“if...else”语法结构。

在这里，我将简单介绍 “if...else” 语法结构，主要目的是让你看懂今天我们这个任务的代码，后续在课程逐步展开的过程中，我还会逐步的引入这个语法结构的一些其他知识点。

我们先来看 “if...else” 最基本的语法结构：

 复制代码

```
1 if (条件表达式) 一条语句 1;  
2 else 一条语句 2;
```

简单来说，if 和 else 都是关键字，代表分支逻辑中的 “如果” 和 “否则”。if 后面跟着的括号里面，需要放一个条件表达式，条件表达式如果成立，程序会执行 “语句 1”，否则就会执行 “语句 2”。下面我来举个例子，你就明白了：

 复制代码

```
1 #include <stdio.h>  
2  
3 int main() {  
4     int a, b;  
5     scanf("%d%d", &a, &b);  
6  
7     if (a == b) printf("a is equal to b!\n");  
8     else printf("a is not equal to b!\n");  
9  
10    return 0;  
11 }
```

这段程序中，首先定义了两个变量 a 和 b，然后通过输入函数（scanf）给变量 a、b 赋值。之后就是重点部分了，根据我们上面所说的，如果 if 后面的条件表达式成立，那么就会输出 “a is equal to b!\n”，否则就会输出 “a is not equal to b!\n”。

最后，我就再带你理解两个概念，一是条件表达式是什么，二是怎样理解 if 后面跟一条语句，所谓一条语句的概念范围是什么。

回到上面的程序中，你会看到程序中的 if 后面跟着一个括号，括号里面放着一个表达式，这个就是我们所谓的条件表达式，而这个括号，是必不可少的。我们发现，这个条件表达式用**两个等号**连接 a 和 b，作用是判断 a 和 b 里面存储的值是否相等。可千万别跟赋值表达式的一个**等号**弄混了。

说到这里，我要告诉你一个重要的事实，变量有变量对应的值，表达式也有表达式对应的值。那么例如上面代码中的条件表达式 “a == b” 所对应的值是什么呢？其实就是数字 1 或者 0，分别表示 “条件成立”（a 与 b 的值相等）和 “不成立”（a 与 b 的值不相等）。

延伸内容：

那么除了条件判等以外，还有哪些条件运算符呢？有判断不等于的 “a != b”，大于的 “a > b”，小于的 “a < b”，大于等于的 “a >= b”，小于等于的 “a <= b”，逻辑非 “!(a > b)”，等价于 “a <= b”。同时多个条件表达式，还可以用逻辑 && 和 || 进行连接，这个后面我再跟你细说。

事实上，if 的括号里面，不仅可以放条件表达式，类似于 “a - b” 这种的表达式，也是可以当做 if 的条件的。

当一般表达式作为条件的时候，if 是怎么执行的呢？很简单，记住：**表达式的值，非 0 即为真**。例如，下面两行代码，效果等价：

 复制代码

```
1 if (a != b) printf("a is not equal to b!\n");
2 if (a - b) printf("a is not equal to b!\n");
```

你会看到，第二行代码中，用 “a - b” 代替 “a != b”，取得了同样的程序运行效果。因此，你只需要重点思考，表达式 “a - b” 什么时候结果非 0 即可，是不是当且仅当 “a != b” 时，“a - b” 的结果非 0，根据之前所说的非 0 即为真，那么 if 条件也就算是成立了。

最后，我们来讲一下怎么理解 “**if 后面跟一条语句**” 这个概念，其实指的是 if 后面的条件成立时所执行的代码。这里，我们的重点是要理解一条语句都包含什么形式，大致可以分为如下几类。

第一种，空语句，就是什么都没有，单纯以一个分号结尾，例如下面这行代码，即使条件成立，也不会有任何实质上的操作。

[复制代码](#)

```
1 if (a == 3) ;
```

第二种，单一语句，比空语句多了语句内容，以分号结尾，例如下面这行代码，当条件成立的时候，会输出 “hello geek!” 。

[复制代码](#)

```
1 if (a == 3) printf("hello geek!\n");
```

第三种，复合语句，被大括号包裹，中间是若干条语句，例如下面这段代码：

[复制代码](#)

```
1 if (a == 3) {  
2     printf("hello geek1!\n");  
3     printf("hello geek2!\n");  
4     printf("hello geek3!\n");  
5 }
```

当条件成立以后，程序会依次执行大括号里面的三条语句：

[复制代码](#)

```
1 hello geek1!  
2 hello geek2!  
3 hello geek3!
```

第四种，结构语句，以 if, for, while 等开头的分支语句或循环语句，例如下面这段代码，首先会先判断 `a==3`，如果条件成立，才会执行下面第二条 if 分支语句，当第二条 if 分支语句的条件也成立的时候，才会输出 “hello geek!” 。

[复制代码](#)

```
1 if (a == 3)  
2     if (b == 4) {  
3         printf("hello geek!\n");  
4     }
```

由此可以看到，if 后面所谓跟着的一条语句，还真是丰富多彩，你可以在后面跟上像上面代码中所写的 printf 函数调用的单一语句，也可以用一个括号，里面放上若干条语句，亦或是 if 后面跟着另一个 if 也是可以的！你看这种组合能力，有没有点儿像乐高玩具？


至此，你就已经掌握了基础的将“如果.....就.....否则.....”这种逻辑结构转换成代码的能力了。你的计算机，终于有了“判断力”。

2. 给程序添加重复执行功能：for 和 while 语句

想想小的时候，你最讨厌什么事情？我最讨厌的就是被老师罚写汉字，错一个字，罚写 100 遍那种的，在我看来真的是在浪费时间。可当我学了程序以后，我发现，程序真的是特别擅长做这种重复的工作，而实现这种功能的语法结构就是 for 语句和 while 语句。

我们先来看语法结构较简单的 while 语句：


```
1 while (循环条件) 一条语句;
```

 复制代码

以 while 关键字开头，后面跟着循环条件，也就是一个条件表达式，然后是一条语句。while 循环，顾名思义，当循环条件成立时，就会执行一次后面的语句，之后就是再判断循环语句是否成立，如果成立就再执行，一直到循环条件不成立为止。

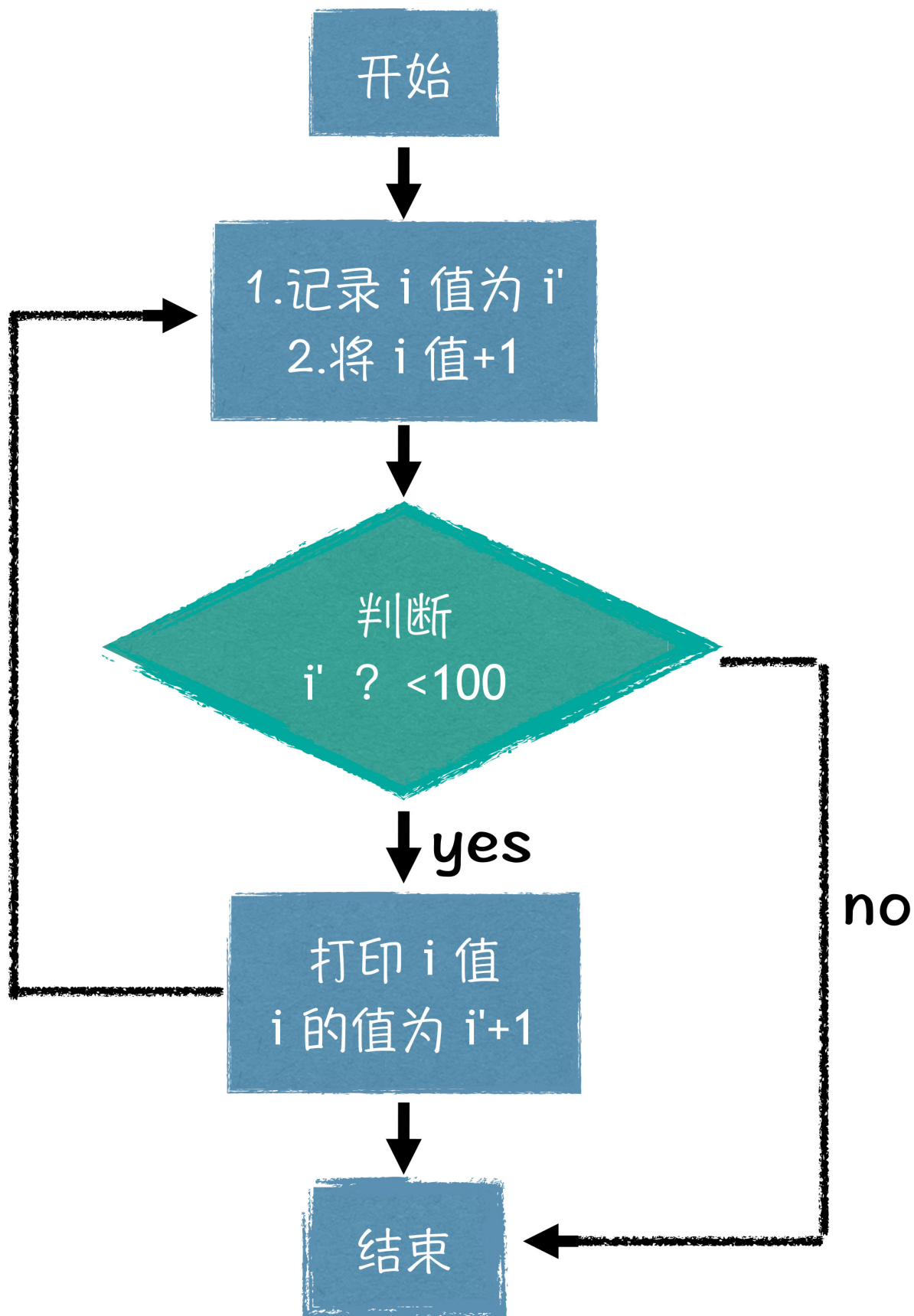
下面呢，我们就用最简单的形式，利用 while 循环，输出前 100 个正整数：

```
1 int i = 0;
2 while (i++ < 100) printf("%d\n", i);
```

 复制代码

这段代码里面，出现了一个你之前没有见过的语法，就是 `i++`，这也是表达式，这个表达式的值等于 `i` 之前的值，当这条表达式执行完以后，`i` 会变成 `i + 1` 的值。例如，起初 `i = 2`，`i++` 表达式的值就等于 2，可表达式执行以后，你要是输出 `i` 的值，这时 `i` 实际等于 3。

上面代码中，我们是用 `i++` 表达式的值和 100 进行比较，表达式的值会遍历 0 到 99 所有的值，由于 `printf` 在 `i++` 之后输出 `i` 的值，所以实际上每次输出的都是 `i + 1` 之后的值，也就是说 `printf` 会输出 1 ~ 100 所有值。具体的你可以参考下面的这个程序流程图。



while 循环流程图

另外，顺便再问你个问题，你还记得上一节课里，我们学到的 `\n` 和 `%d` 分别代表什么意思嘛？如果不记得，记得回去再复习下。

有了 `while` 循环语句的加持之后，是不是重复做某件事，变得很方便了呢？不急，下面我要给你介绍的是功能更为强大的 `for` 语句。还是先来看一下 `for` 语句的结构吧：

```
1 for (初始化①; 循环条件②; 循环后操作③) 一条语句④;
```


 复制代码

正如你看到的，我把 `for` 语句的四部分已经给你标出来了，`for` 语句会按照 ①②④③②④③... 循环，直到某一次循环条件②不成立了为止。

你会发现，①这一部分只在循环开始时执行了一次，真正所谓的循环，是以循环条件②，一条语句④以及循环后操作③组成的。

如果要是用 `for` 循环输出 1 ~ 100 所有值，会显得代码更清晰一些：

```
1 for (int i = 1; i <= 100; i++) printf("%d\n", i);
```

 复制代码

上面这段代码，就是用 `for` 循环实现了和之前 `while` 循环相同的功能。

看了 `for` 循环和 `while` 循环以后，你可能会问，实际中哪种循环用的比较多，我个人经验来说，`for` 循环用的比较多，因为 `for` 循环每一部分都非常明确，对于比较复杂的循环控制过程，`for` 循环写出来以后，一般都会比 `while` 循环可读性强。

为了让你感受到 `for` 循环真正的威力，写一段代码，让你感受一下：

```
1 for (int i = 1, k = 0; i <= 48; i++, k += 2) printf("%d\n", k);
```

 复制代码

上面这段程序中，我们用到了两个同步信息变量，`i` 和 `k`，`i` 从 1 到 48，保证循环了 48 次；代码中 “`k+=2`” 表示 `k` 每次增加 2，也就是说，在这个过程中，`i` 遍历了 1 到 48 这

48 个整型值，而 k 同步地遍历了从 0 开始的前 48 个偶数。这段代码的意思其实就是打印出从 0 开始后的共 48 个偶数，即 0、2、4.....92、94。

如果用 while 来实现这个目的，知道怎么写吗？你可以自己在计算机上试一下。


一起动手，搞事情

思考题：打印乘法表

使用循环和条件判断，打印一个格式优美的 66 乘法表

要求 1：输出内容及样式参照下面给出的样例

要求 2：每两列之间用 \t 字符进行分隔，行尾无多余 \t 字符

 复制代码

```
1  1*1=1
2  1*2=2  2*2=4
3  1*3=3  2*3=6  3*3=9
4  1*4=4  2*4=8  3*4=12  4*4=16
5  1*5=5  2*5=10  3*5=15  4*5=20  5*5=25
6  1*6=6  2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
```

“日期计算器”程序完成

准备完了所有的基础技能后，就让我们来完成开始说的那个任务吧，我们来思考一下哈，首先我们需要有一个循环，循环每一次，让计算机帮我们计算一次下一天的日期。每次在计算下一天日期的过程中，先让日子加 1，判断是否跨月，如果跨过了一个月份，就让日子从 1 开始，让月份加 1，再判断是否跨年，如果跨年了，就让月份从 1 开始，年份加 1。

如上的过程中，有一个关键问题需要你注意，就是 2 月份的月份天数的计算方法，咱们来简单回顾一下闰年的规则，年份满足以下其中一条即为闰年：

能被 4 整除，但不能被 100 整除；

能被 400 整除。

如果把闰年的规则翻译成逻辑判断，应该是下面这个样子：

```
1 if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) ...
```

[📄 复制代码](#)

下面就让我们把思路过程转换成程序过程：

[📄 复制代码](#)

```
1 #include <stdio.h>
2
3 int main() {
4     int y, m, d, X; // 定义存储 年月日 和 X 的变量
5     scanf("%d%d%d", &y, &m, &d); // 读入年月日
6     scanf("%d", &X); // 读入 X 值
7     for (int i = 0; i < X; i++) { // 循环 X 次, 每次向后推一天
8         d += 1;
9         switch (m) {
10             case 1:
11             case 3:
12             case 5: { // 第一部分逻辑
13                 if (d > 31) d = 1, m += 1;
14                 if (m == 13) m = 1, y += 1;
15             }; break;
16             case 4:
17             case 6: { // 第二部分逻辑
18                 if (d > 30) d = 1, m += 1;
19             } break;
20             case 2: { // 第三部分逻辑
21                 if ((y % 4 == 0 && y % 100 != 0) || y % 400 == 0) {
22                     if (d > 29) d = 1, m += 1;
23                 } else if (d > 28) {
24                     d = 1, m += 1;
25                 }
26             } break;
27         }
28     }
29     printf("%d %d %d\n", y, m, d);
30     return 0;
31 }
```

上面这段程序是个半成品，只处理了前 6 个月的情况，并且用到了 **switch...case 的分支结构**，与 if 结构类似，都是用于做逻辑分支判断的。关于这部分的内容，给你留个小作业，自学一下 switch...case 分支结构，然后按照自己的理解，补全上述程序，使得上述程序可以处理一年中 12 个月的全部情况。

虽然这个程序中有一部分内容需要你进行自学，可你也不要担心，我还是会跟你详细解释上述程序设计的思路。读入部分的代码，相信你现在已经可以很好的掌握了，这一部分就不展开解释了。程序整体设计中，是用 for 循环包裹了 switch...case 结构，for 循环负责循环 X 次，每次在循环内部，都将对日子变量 d 进行加 1 操作，而在 switch...case 结构内部，主要是处理跨月和跨年的问题。

你会看到 switch...case 结构中，主要分成三部分逻辑，第一部分逻辑，主要处理天数为 31 天的月份，由于 12 月也是 31 天，所以当本月是 12 月，并且发生了跨月，变成了 13 月，说明是到了下一年的 1 月，需要将年份 +1，月份置为 1 月。第二部分逻辑，主要处理天数为 30 天的月份。第三部分逻辑，主要处理 2 月份的情况，在这里，程序中分成两种情况来讨论，闰年和非闰年，闰年的时候，判断日子是否超过 29 天，非闰年，判断日子是否超过 28 天。

我保证，在你尝试补全上述程序的过程中，你会发现，上述程序易于修改和补全，你要是能试着将上述程序修改成 if 分支结构，那就更好了。这样你将对上述程序结构的美，会感受的更深刻。

课程小结

最后呢，来总结一下今天所学的重点。今天呢，我们主要学习了两种程序流程控制结构，一种分支结构，主要以 if 语句为代表，另一种循环结构，以 for 循环和 while 循环为代表。如果说你只想记住几点的话，那么应该是以下几点：

1. 熟练掌握分支和循环结构的执行顺序，这一点很重要。
2. if 语句，首先判断条件表达式的真假，如果为真，则执行 if 里面的语句。
3. for 循环，分成四部分，其中②④③部分，构成了一个循环，第①部分是用做初始化的。
4. 所谓一条语句的概念，包括了空语句，单一语句，复合语句和结构语句。

以上这 4 点要牢记哦，尤其是其中的分支和循环结构的执行顺序，因为掌握和理解了程序的执行顺序，才是分析程序，理解程序的第一步。

好了，今天就到这里了，下期我将带你来做一个小总结，我将带你学习一个有趣的圆周率的计算方法，我是胡光，我们下期见。

人人都能学会的编程入门课

>>> 每天 10 分钟，轻松学编程

胡光

原百度高级算法研发工程师



新版升级：点击「 请朋友读」，20位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 02 | 第一个程序：教你输出彩色的文字

下一篇 04 | 随机函数：随机实验真的可以算 π 值嘛？

精选留言 (4)

写留言



Geek_And_Lee00

2020-01-13

老师，我的问题是：什么都没有输出。编译没有错误，老师有时间了麻烦帮忙看下。代码如下：

```
#include <stdio.h>
```

...

展开

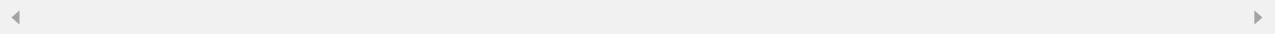


Paradise

2020-01-13

思路清晰，语言干练，通俗易懂，做了几年前端看老师的文章一样觉得很有收获！

作者回复: d(^_^o)



徐洲更

2020-01-09

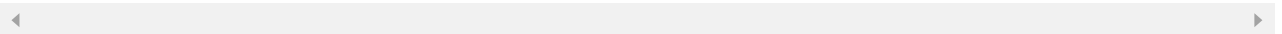
作业地址: <https://github.com/xuzhougeng/learn-c-in-JKSJ/blob/master/jksj-c-03.c>

我今天写作业的时候，明明觉得自己的逻辑好像是对的，但是结果错了，应该如何debug呢？特别是写了那么长的代码的时候。

展开 ▾

作者回复: 看到你的代码了，对于 switch...case 的整体结构理解很准确。debug 的话，可以尝试如下三种方式：

1. 心算推理法，通过看代码找 bug，这个通常适用于经验丰富的老手。
2. 关键步骤输出信息，在程序中，找到关键的位置，输出相关变量的值，通过程序输出找错误。
3. 借助调试工具，IDE 一般都带有自己的单步调试工具，Linux 下面可以借助 gdb 进行调试。



行问

2020-01-09

请教下，C 语言的编码规范有什么推荐的，专栏中的代码代码规范是如何进行的？

作者回复: C语言的专门的编码规范在国内不太多，基本都是参考一些流行的C++的编码规范。

