

# CPL 第五次编程练习 5-function 题解

教学周历中本周课程的知识为：

函数的概念与使用；作用域与程序结构

教材章节：

9.1 - 9.5、10.1 - 10.5

## 必做题 (behaviors.c)

本题知识点：

关于抄袭、常见错误和提问的情况总结

我们逐条分析：

ant-hengxin 复制了 StackOverflow/CSDN 上的代码提交到 OJ 网站上并且 AC 了；

✗显然犯了抄袭的大罪。

ant-hengxin 在作业截止之前把自己的 AC 代码上传到了 github 的公开仓库；

✗显然违反了学术诚信的供他人抄袭的错误。

DDL 马上就要到了，ant-hengxin 的题做不完了，他就算空着不做也绝不抄袭；

✓你做得好啊！（但大家还是要争取做完，毕竟每天都有线下答疑）

ant-hengxin 需要一个输入长度为  $n$  的字符串，他这么写：

```
1 | int n;  
2 | scanf("%d", &n);  
3 | char str[n + 2];  
4 | scanf("%s", str);  
5 | // other codes
```

✗不要使用可变长数组！！你会错得很惨！

ant-hengxin 需要开一个  $10^8$  长度的 double 数组，他这么写：

```
1 | int main(void) {  
2 |     double array[100000000];  
3 |     // other codes  
4 | }
```

✗在函数（主函数）内部开不了这么大的数组，代码无法运行。（可以自己尝试上限是多少）

ant-hengxin 需要为长度为  $10^3$  的字符串开数组，他这么写：

```
1 | char str[1000];
```

✗ ? 字符串建议不要卡着上限大小开，应尽量留出几位空位，防止运行错误。（数组越界）

ant-hengxin 需要开一个  $10^5$  的 int 数组，他这么写：

```
1 | int nums[10005];
```

✗ 十的五次方后面究竟有几个零？下次再有人把  $10^5$  写成 10005 直接封号怎么样？

ant-hengxin 帮 shuilongzhihun debug，他找不出来哪里有问题，于是复制了 shuilongzhihun 的代码，改动了一点，并用自己的账号交了一发；

✗ 今天你交别人代码，明天查重就到你家。

ant-hengxin 帮 Tilnel debug，他找不出来哪里有问题，于是直接把自己的代码丢给 Tilnel；

✗ 显然犯了提供抄袭的大罪。

ant-hengxin 在群里问：“F 题运行错误是怎么回事？”；

✗ 运行错误无非就那几种情况，具体在课程网站上都有，如数组越界、访问数组[-1]的值、return 1，不要问这种“别人除了把手册扔给你，其他也帮不了你”的问题。

ant-hengxin 在群里问：“D 题 73 分是怎么回事？”；

✗ 虽然错误分数相同可能是共性问题，但更多的情况是，代码所错的点都不一样，用分数来问问题只会：1. 别人没法正常回答你，只能靠猜；2. 自己会浪费更多时间去看别人指出的可能的问题（却不是你的问题所在）

ant-hengxin 直接贴了一段代码，并且问“我本地都是对的，为啥 OJ 说我这个错了啊？”。

✗ 这种问题。。还是多看看自己代码里写了多少未定义行为吧。建议在用gcc跑代码的时候加上 -Wall -O2 来使控制台可以输出报错，并且与 OJ 对齐。

如 gcc test.c -o test -Wall -O2。

---

还有几位同学没答满12题但在第一天通过了.....（确实是 OJ 的问题，但后来同学没改）已经大发慈悲把这几位同学的分加上了。至于没做这题的同学，应该会从作业总分里倒扣100分吧.....

## 求和 (sum.c)

本题知识点：

整数输入、整数运算、循环、函数

简单题，需要理解一下题意。容易发现

$$\underbrace{tt \cdots t}_{n \text{ 个 } t} \cdot 10 + t = \underbrace{tt \cdots t}_{n+1 \text{ 个 } t}$$

只要按照上述关系利用前一项递推计算后一项，这种方法在前几周的作业里出现了很多次，因此本题可以看作前几周作业的简单复习题。

可以看出所有的数据包括中间答案都在 int 范围内。

```

1  #include <stdio.h>
2
3  int main(void) {
4      int n, t, sum = 0, temp = 0;
5      scanf("%d", &n, &t);
6      while (n--)
7          sum += (temp = temp * 10 + t);
8      printf("%d", sum);
9      return 0;
10 }

```

当然，也可以将  $\underbrace{tt \cdots t}_{n \uparrow t}$  展开成  $\sum t \cdot 10^x$  的形式求和计算，代码略。

## 绝对素数 (absolute-prime.c)

本题知识点：

整数输入、整数运算、素性检测、条件分支、循环、函数

枚举每个  $i \in [2, n] \cap \mathbf{Z}$ ，分别检验其是否为素数以及其“翻转”过来之后的数是否为素数；其中，检验是否为素数的部分代码被多次使用，我们强烈建议将其封装成一个函数。

```

1  #include <stdio.h>
2
3  int reverse(int n) {
4      int ans = 0;
5      while (n) { // 翻转数字
6          ans = ans * 10 + n % 10;
7          n /= 10;
8      }
9      return ans;
10 }
11 int check(int n) {
12     for (int i = 2; i * i ≤ n; i++)
13         if (n % i == 0) // 如果有因式，直接返回 “非素数”
14             return 0;
15     return 1;
16 }
17 int main(void) {
18     int n, ans = 0;
19     scanf("%d", &n);
20     for (int i = 2; i ≤ n; i++) {
21         if (check(i) && check(reverse(i)))
22             ans++;
23     }
24     printf("%d", ans);
25     return 0;
26 }

```

## 确定进制 (radix.c)

本题知识点:

整数输入、整数运算、条件分支、循环、n 进制数的性质、函数

在经历了 **N进制转换 (decimal.c)** 的洗礼后, 相信各位同学都对进制有了更深的理解。(尤其是进制不合法的情况)

这道题的思路就是从小到大枚举每个  $B \in [2, 40] \cap \mathbf{Z}$  检验进制是否合法且该算式是否成立, 因为我们要输出最小的合法的  $B$ , 所以找到第一个合法的  $B$  后就直接输出并结束程序; 最后别忘了在枚举结束后返回 0 表示所有的  $B$  都不合法。

需要注意的是, 判断进制合法与判断算式成立是有先后顺序的, 有一些同学犯了一个错误: “找到一个成立的算式, 但进制不合法, 所以输出 0。”这显然是会漏掉合法且成立的进制数的。

由于本题只可能输入不超过两位的数, 使用简单的除法和取模找出最大的那个数位即可。

```
1  #include <stdio.h>
2
3  int legal(int p, int q, int r, int i) {
4      return p % 10 ≥ i || p / 10 ≥ i || q % 10 ≥ i ||
5             q / 10 ≥ i || r % 10 ≥ i || r / 10 ≥ i;
6  }
7
8  int radix(int p, int q, int r) {
9      for (int i = 2; i < 41; i++) {
10         if (legal(p, q, r, i))
11             continue;
12         int P = p % 10 + (p / 10) * i;
13         int Q = q % 10 + (q / 10) * i;
14         int R = r % 10 + (r / 10) * i;
15         if (P * Q == R)
16             return i;
17     }
18     return 0;
19 }
20
21 int main(void) {
22     int p, q, r;
23     scanf("%d %d %d", &p, &q, &r);
24     printf("%d", radix(p, q, r));
25     return 0;
26 }
```

## 下一个排列 (next-permutation.c)

本题知识点:

整数输入、排序、排列、循环、条件分支、整数一维数组的初始化与访问, 留一个思考题 (证明)

这是一道阅读理解题, 同学们需要读懂提示中的信息, 实现给出的方法。我在写题面的时候对于“最长递降后缀”的“最长”和“递降”都做出了解释, 唯独没有解释“后缀”, 因此造成了部分同学理解上的不便。“后缀”是指一定包括序列结尾的数的“子串”(翻译于 substring, 区别于 subsequence (子序列), 后者可以不要求其中的数连续), 举个例子, **1 4 3 2 5** 的后缀有 **5**、**2 5**、**3 2 5**、**4 3 2 5**、**1 4 3 2 5**。

读懂了提示之后, 大家只需要按照题意实现代码, 需要三步:

- 首先找到最长递降后缀的长度, 并得到其前一个数。

- 交换前一个数与最长递降后缀中大于它的那个最小的数。
- 将后缀长度的位置的数翻转。

```

1  #include <stdio.h>
2
3  int n, a[2005];
4
5  int main(void) {
6      scanf("%d", &n);
7      for (int i = 1; i ≤ n; i++) scanf("%d", &a[i]);
8      int p = n;
9      while (a[p - 1] > a[p]) p--;
10     for (int i = n; i ≥ p; i--)
11         if (a[p - 1] < a[i]) {
12             a[p - 1] ^= a[i] ^= a[p - 1] ^= a[i]; // 交换
13             break;
14         }
15     for (int i = p, j = n; i < j; i++, j--)
16         a[i] ^= a[j] ^= a[i] ^= a[j]; // 交换
17
18     for (int i = 1; i ≤ n; i++) printf("%d ", a[i]);
19     return 0;
20 }
```

事实上标程很短，写得长的同学可以参考一下。

## 幻方 (magic-square.c)

本题知识点：

整数输入、条件分支、循环、整数二维数组的初始化与访问

阅读理解题 ×2，大家需要阅读理解题目中给出的构造奇数阶幻方的方法并输出。

这里我们采用的方法是用两个变量  $x$  和  $y$  记录当前填到位置的坐标，每步操作需要做两件事情：

- 填入当前的数  $i$ 。
- 移动  $x$  和  $y$  到下一个需要填数的位置，顺便检查  $x$  和  $y$  的合法性并完成调整。

我们将  $x = 1, y = \frac{(n+1)}{2}$  作为第一个位置，容易发现循环  $n^2$  次后，所有的数都已经填好，此时只要进行输出就可以了。

关于位置合法性的检查，相信数学好的同学会发现，如果我们将矩阵的下标记为

$([0, n - 1] \cap \mathbf{Z}) \times ([0, n - 1] \cap \mathbf{Z})$  的话，对于那些出界的  $x$  和  $y$  可以直接对  $n$  进行取模。如果你没有发现，那也没有关系，你可以写一些条件语句判断一下达到同样的效果。

采用条件语句判断的方法如下。

```

1  #include <stdio.h>
2
3  int a[1005][1005];
4  int main() {
5      int n;
6      scanf("%d", &n);
7      int x = (1 + n) / 2, y = 1;
8      for (int i = 1; i ≤ n * n; i++) {
9          a[x][y] = i;
```

```

10     if (x == n && y == 1) {
11         y = 2;
12         continue;
13     }
14     if (x == n) {
15         y--;
16         x = 1;
17         continue;
18     }
19     if (y == 1) {
20         x++;
21         y = n;
22         continue;
23     }
24     if (a[x + 1][y - 1] != 0) {
25         y++; continue;
26     } else {
27         x++, y--; continue;
28     }
29 }
30 for (int i = 1; i ≤ n; i++) {
31     for (int j = 1; j ≤ n; j++) {
32         printf("%d ", a[i][j]);
33     }
34     printf("\n");
35 }
36 return 0;
37 }

```

采用直接取模的方法如下。

```

1  #include <stdio.h>
2
3  int n, ans[1005][1005];
4
5  int main() {
6      scanf("%d", &n);
7      int x = 0, y = (n - 1) / 2;
8      for (int i = 1; i ≤ n * n; i++) {
9          ans[x][y] = i;
10         int nx = (x - 1 + n) % n, ny = (y + 1) % n;
11         if (ans[nx][ny]) {
12             nx = (x + 1) % n;
13             ny = y;
14         }
15         x = nx;
16         y = ny;
17     }
18     for (int i = 0; i < n; i++)
19         for (int j = 0; j < n; j++)
20             printf("%d%c", ans[i][j], " \n"[j + 1 == n]);
21     // " \n"[j + 1 == n] 这是一种新奇的输出空格or换行的方式，大家自行领悟
22     return 0;
23 }

```