

PTA 选择结构 H

rogeryoung

2021 年 04 月 13 日

目录

1	阶梯付费	2
1.1	7-15 超市购物打折	2
1.2	7-18 计算出租车费	3
2	向上整除	3
2.1	7-7 计算邮资	3
3	利用数组	4
3.1	7-19 小燕爱偶数	4
3.2	7-22 前天是哪天	5
4	细节处理	6
4.1	7-1 今天之前的第 n 天是星期几	6
4.2	7-9 日期识别 2	7
4.3	7-12 位置关系 A	9
5	附录·私货	10
5.1	关于未定义行为	10
5.2	后续学习	11
5.3	求值顺序测试	11

PTA 选择结构 HARD 部分, [PDF](#)。

1 阶梯付费

对于这种阶梯付费的问题，每一档可以记作：超过 x_i 元时，超出部分费率 k_i 。于是我们可以令 f_i 为超出价格，有

$$f_i = \begin{cases} p - x_i & , p > x_i \\ 0 & , p \leq x_i \end{cases}$$

那么总付费可以表示为 $\sum k_i f_i$ 。

1.1 7-15 超市购物打折

```
1  int main() {
2      double t;
3      scanf("%lf", &t);
4      double f1, f2, f3;
5      f1 = f2 = f3 = 0;
6      if (t ≤ 50) {
7          f1 = t;
8      } else {
9          f1 = 50;
10         t -= 50;
11         if (t ≤ 100) {
12             f2 = t;
13         } else {
14             f2 = 100;
15             t -= 100;
16             f3 = t;
17         }
18     }
19     printf("%.2lf", f1 + f2*.9 + f3*.8);
20     return 0;
21 }
```

1.2 7-18 计算出租车费

题目有坑，当 $x = 0$ 时应当付 0 元。

```
1 int main() {
2     double x;
3     scanf("%lf", &x);
4     if (x == 0) {
5         printf("0.0");
6         return 0;
7     }
8     double f1, f2;
9     f1 = f2 = 0;
10    if (x > 3)
11        f1 = x - 3;
12    if (x > 10)
13        f2 = x - 10;
14    printf("%.1lf", 11.0 + f1*2.4 + f2*0.96);
15    return 0;
16 }
```

2 向上整除

我们知道，C 语言中的除法是向下取整。

$$p/q = \left\lfloor \frac{p}{q} \right\rfloor$$

于是向上取整可以表示为

$$\left\lceil \frac{p}{q} \right\rceil = \left\lfloor \frac{p+q-1}{q} \right\rfloor = \left\lfloor \frac{p-1}{q} \right\rfloor + 1 = (p-1)/q + 1$$

如果使用 `if` 判断整除的写法，对于分母为 1 的特例很可能出错，我展示的写法兼容这种特例。

2.1 7-7 计算邮资

即金额是对 500 向上整除。

```
1 int main() {
2     int n;
3     char c;
4     scanf("%d %c", &n, &c);
5     int ans = 8;
6     if (n > 1000) {
7         n = n - 1000;
8         if (n < 0)
9             n = 0;
10        n = (n - 1) / 500 + 1;
11        ans += n * 4;
12    }
13    if (c == 'y')
14        ans += 5;
15    printf("%d", ans);
16    return 0;
17 }
```

3 利用数组

一部分题其实 `if - else` 或 `switch` 都能做，我只是觉得数组更适合。

3.1 7-19 小燕爱偶数

使用数组 a 记录所有的偶数， p 是数组的下标。

当遇到偶数 t 时，令 $a_p = t$ ，自增 p 。

这样可以实现不断放入，令 i 遍历 $0 \rightarrow p$ 输出 a_i 即可。

不要忘了末尾没有空格。

```
1 int a[10086], p = 0;
2
3 int main() {
4     int n;
5     scanf("%d", &n);
```

```

6     for (int i = 0; i < n; i++) {
7         int t;
8         scanf("%d", &t);
9         if (t % 2 == 0)
10            a[p++] = t;
11    }
12    printf("%d\n", p);
13    if (p > 0)
14        printf("%d", a[0]);
15    for (int i = 1; i < p; i++)
16        printf(" %d", a[i]);
17    return 0;
18 }

```

3.2 7-22 前天是哪天

把当月天数存到数组里，特判闰年。

需要退位时，取出天数计算一下。

```

1  int month[] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
2
3  int main() {
4      int y,m,d;
5      scanf("%d %d %d",&y,&m,&d);
6      d -= 2;
7      if (d ≤ 0) {
8          m--;
9          if (m ≤ 0) {
10             y--;
11             m = 12;
12         }
13         int day = month[m];
14         int flag = (y % 4 == 0 && y % 100 ≠ 0);
15         if (m == 2)
16             if (y % 400 == 0 || flag)

```

```

17         day++;
18         d = day + d;
19     }
20     printf("%d-%d-%d", y, m, d);
21     return 0;
22 }

```

4 细节处理

这里的题需要些许思考。

4.1 7-1 今天之前的第 n 天是星期几

求星期 x 的 n 天前是星期几，朴素的想法是对 x 自减 n 次，特判当 $x = 1$ 时后继是 7。

```

1  int main() {
2      int x, n;
3      scanf("%d %d", &x, &n);
4      for (int i = 0; i < n; i++) {
5          x--;
6          if (x == 0)
7              x = 7;
8      }
9      printf("%d", x);
10     return 0;
11 }

```

很容易想到利用取模来优化。为了方便，将星期一到星期天映射到 $0 \sim 6$ 。那样第 $x - n$ 天是星期几，完全只和其对 7 的余数有关。

唯一的问题是负数取模，它的结果可能与你的想象不同，感兴趣的可以查看 [cp-preference](#)。我们可以考虑避免它，容易验证

$$x - n \bmod 7 \equiv x - n \pmod{7}$$

上式左侧显然大于 -7 ，于是有

```
1 int main() {
2     int x, n;
3     scanf("%d %d", &x, &n);
4     n = n % 7;
5     x--;
6     x = (x - n + 7) % 7;
7     printf("%d", x + 1);
8     return 0;
9 }
```

或者，我们还有另一种常见的写法 $(a \% b + b) \% b$ ，完整如下

```
1 int main() {
2     int x, n;
3     scanf("%d %d", &x, &n);
4     x--;
5     x = ((x - n) % 7 + 7) % 7;
6     printf("%d", x + 1);
7     return 0;
8 }
```

4.2 7-9 日期识别 2

本题略要思考。首先因为分隔符未知，可以使用 `%c` 吸收掉，进行 6 次判断。

```
1 int main() {
2     int a,b,c;
3     char p;
4     scanf("%d%c%d%c%d", &a, &p, &b, &p, &c);
5     int flag = 0;
6     flag += test(a, b, c);
7     flag += test(b, c, a);
8     flag += test(c, a, b);
9     flag += test(c, b, a);
10    flag += test(a, c, b);
11    flag += test(b, a, c);
```

```

12     if(flag)
13         printf("%d", flag);
14     else
15         printf("Invalid Date!\n");
16     return 0;
17 }

```

对于合法的年月，再计算当前月份应有多少天，完全符合返回 1，不符合则返回 0。

```

1  int month[] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
2
3  int test(int y, int m, int d) {
4      if(0 ≤ y && 1 ≤ m && m ≤ 12) {
5          int day = month[m];
6          int run = y % 400 == 0 || (y % 4 == 0 && y % 100 ≠ 0);
7          if (m == 2 && run)
8              day++;
9          if(0 < d && d ≤ day)
10             return 1;
11     }
12     return 0;
13 }

```

当一行代码过长时，尽量分开写。过长的代码往往意味着复杂的逻辑，分开有助于理解。

也可以自己实现读入数字，示范如下。一般用这个读数字比 scanf 快的多，也叫做快读。

```

1  int read() {
2      int s = 0;
3      char c = getchar();
4      while (c ≥ '0' && c ≤ '9') {
5          s = s * 10 + c - '0';
6          c = getchar();
7      }
8      return s;

```


4.3 7-12 位置关系 A

比较有趣的数学题。定义圆心距 d 为

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

将 d 与 $|r_1 + r_2|$ 和 $|r_1 - r_2|$ 比较, 分类讨论如下

- 当 $|r_1 + r_2| < d$, 两圆尚未接触, 为外离 Separated。
- 当 $d = |r_1 + r_2|$, 两圆外切 Circumscribed。
- 当 $|r_1 - r_2| < d < |r_1 + r_2|$, 两圆相交 Intersected。
- 当 $d = |r_1 - r_2|$, 两圆内切 Inscribed。
- 当 $0 \leq d < |r_1 - r_2|$, 两圆内含 Contained。

至于重合, 开始特判掉即可。

```
1 int main() {
2     int x1, y1, r1, x2, y2, r2;
3     scanf("%d %d %d", &x1, &y1, &r1);
4     scanf("%d %d %d", &x2, &y2, &r2);
5
6     int td = (x1-x2) * (x1-x2) + (y1-y2) * (y1-y2);
7     int tr1 = (r1-r2) * (r1-r2);
8     int tr2 = (r1+r2) * (r1+r2);
9
10    if (td == 0 && tr1 == 0) {
11        printf("Completely Overlapping");
12    } else if (td < tr1) {
13        printf("Contained");
14    } else if (td == tr1) {
15        printf("Inscribed");
16    } else if (td < tr2) {
17        printf("Intersected");
18    } else if (td == tr2) {
19        printf("Circumscribed");
```

```
20     } else {
21         printf("Separated");
22     }
23     return 0;
24 }
```

5 附录·私货

其实我正文里写了不少私货了（

5.1 关于未定义行为

未定义行为是很重要的概念。不知道你是否为这些代码的值苦恼过？

```
1 i = ++i + i++;
2 i = i++ + 1;
3 f(++i, ++i);
```

未定义行为的含义是，出现什么结果都不奇怪，比如像段错误，程序闪退，电脑死机，系统崩溃，都有可能。甚至在两个电脑上结果不同，都不奇怪。

C 语言并不是教条主义的语言，它是先推广开才进行标准化的，因此 C/C++ 标准都是追述性的。编译器也是程序，我们需要给编译器充足的空间去进行代码优化，而不是做很多无用而详细的限制。

比如运算符的优先级和求值顺序无关，求值顺序是由 **序列点** 概念决定，序列点未指定的则不指定顺序。这属于实现编译器才需要了解的知识，我们没必要了解。我们总是可以通过把程序写的简炼，来避免通过复杂的知识确定程序的运行状况。

任何良好的程序不应该依赖未定义行为，甚至不应该使用不常见的写法。很多复杂的概念，你在编程里用了，除了把自己绕晕以外什么用也没有。但是就是有人以绕晕别人为荣，天天琢磨茴的四种写法，遇到问题最喜欢引经据典，我们一般讽刺作“语言律师”。

说了这么多，一言以蔽之：

不要写看不懂的代码。

5.2 后续学习

C 语言受底层影响很深。比如 `switch` 需要自己写 `break`，当了解到跳转表 Jump Table 后，反倒觉得很自然了。

归根结底，奇怪的不是 C 语言，而是 CPU 就是那样工作的。缺乏理解时，编程就像念咒语。

估计很多同学 C 语言学的差不多了吧，我介绍一种后续的学习方案。

很多学校在大一的程序设计之后会开一门计算机系统导论 (ICS)，使用的教材是《深入理解计算机系统》(CS:APP)。这门课引进自国外的 CMU15-213，讲述了以程序员的视角（相对的是系统设计视角）需要了解的计算机系统。读完后，你会去开始思考代码的底层运作原理，而不是把计算机当做黑盒。

这门课只是一门导论，带你领略整个计算机系统的宏观面貌，其细节将会在之后组成原理、操作系统、体系结构、网络等课程深入。

网上也有诸多资料，可以配合学习。

5.3 求值顺序测试

求值顺序是由 **序列点** 概念决定，序列点未指定的则不指定顺序。

为了查看求值顺序，我们测试了如下程序

```
1  int x;
2
3  int f() {
4      int t;
5      scanf("%d", &t);
6      x = t;
7      return t;
8  }
9
10 int main() {
11     int ans = f() + f() * f();
12     printf("%d %d", ans, x);
13     return 0;
14 }
```

当你输入 1 2 3 时，此时 `vc6`，`gcc`，`clang` 都很统一的告诉你输出会是 7 3，这与我们的想法不符。这是值得信赖的规律吗？

我们可以让它更混乱一点

```
1 int main() {  
2     printf("%d %d %d", f(), f(), f());  
3     return 0;  
4 }
```

还是输入 1 2 3, 答案就有很多种了

```
1 gcc    : 3 2 1  
2 tcc    : 1 2 3  
3 vc6    : 3 2 1  
4 clang : 1 2 3
```

标准中明确指出了, 序列点之间的副作用顺序未指定, 因此这部分实验没有任何意义。