

# PTA 循环结构 E

rogeryoung

2021 年 05 月 08 日

## 目录

<b>1</b>	<b>约定</b>	<b>2</b>
<b>2</b>	<b>时间复杂度</b>	<b>2</b>
<b>3</b>	<b>简单循环</b>	<b>3</b>
3.1	7-1 输出等腰杨辉三角	3
3.2	7-2 含 8 的数字的个数	4
3.3	7-3 立方和	4
3.4	！ 7-4 统计单词数量	5
3.5	7-6 中国余数定理	6
3.6	7-7 多项式求值	6
3.7	7-9 英文字母替换加密	7
3.8	7-10 字母塔	7
3.9	7-12 质因子分解	7
3.10	7-13 启程几何	8
3.11	7-14 进度条	9
3.12	7-17 计算到任意日期的总天数	9
3.13	7-18 最大值和最小值	10
3.14	7-19 计算评分	10
3.15	7-20 我们爱运动	10
3.16	7-21 累加 a-aa+aaa-aaaa+...	11
3.17	7-23 循环 - n 个数最大值	11
3.18	7-25 求简单交错序列的前 N 项和	12
3.19	7-26 谁最高	12
<b>4</b>	<b>格式输出</b>	<b>13</b>
4.1	7-4 输出 2 到 n 之间的全部素数	13
4.2	7-5 输出前 n 个 Fibonacci 数	14
4.3	7-11 数字菱形	15
4.4	7-15 重排矩阵	16
4.5	7-16 1000 以内所有各位数字之和为 n 的正整数	18
4.6	7-22 嵌套循环-素数的和	19

PTA 循环结构 EASY 部分, [PDF](#)。

## 1 约定

随着程序逐渐复杂, 我决定介绍一些算法竞赛中常用的定义, 来简化我们的程序。

---

```
1 typedef long long ll;
2 #define _fora(i,a,n) for(int i=(a);i<=(n);i++)
3 #define _forz(i,a,n) for(int i=(a);i>=(n);i--)
```

---

即 `ll` 是 `long long` 类型的简写, 能够带来比 `int` 更大的范围。

同样, `_fora` 和 `_forz` 是 `for` 的简写。

可能会在讲解中使用它们, 感兴趣的可以尝试。

## 2 时间复杂度

可能已经有同学察觉到, 尽管很多写法都可以通过测试, 可它们并不是一样快的。

比如, 求

$$\sum_{i=1}^n \sum_{j=1}^i j = \sum_{i=1}^n \frac{i(i+1)}{2} = \frac{1}{6}n(n+1)(n+2)$$

我们可以写出四种写法

---

```
1 int sum0(int n) { // 不会真有人这么写吧
2     int sum = 0;
3     for (int i = 1; i <= n; i++)
4         for (int j = 1; j <= i; j++)
5             for (int k = 1; k <= j; k++)
6                 sum++;
7     return sum;
8 }
9
10 int sum1(int n) {
11     int sum = 0;
12     for (int i = 1; i <= n; i++)
13         for (int j = 1; j <= i; j++)
14             sum += j;
15     return sum;
16 }
17
18 int sum2(int n) {
19     int sum = 0;
```

```

20     for (int i = 1; i <= n; i++)
21         sum += i * (i + 1) / 2;
22     return sum;
23 }
24
25 int sum3(int n) {
26     return n * (n + 1) * (n + 2) / 6;
27 }

```

可以得出，sum0 累加执行的次数是  $\frac{1}{6}(n^3 + 3n^2 + 2n)$ ，sum1 累加执行的次数是  $\frac{1}{2}(n^2 + n)$ ，而 sum2 需要  $n$  次运算，sum3 只需要 1 次。

尽管电脑的运行速度很快，可并不是无限快，不同的算法所需时间差别可能会很大。

当  $n$  较大时，比如  $n = 10000$ ，sum0 需要接近  $10^{12}$  次，sum1 大致  $10^8$  次，sum2 大致  $10^4$ ，而 sum3 还是 1 次，差距非常明显。

为了凸显算法的运行时间和  $n$  的关系，在表示算法的时间复杂度时常常略去常数和低阶项，系数也会被忽略。比如

$$\frac{1}{6}(n^3 + 3n^2 + 2n) \sim \frac{1}{6}n^3 \sim n^3$$

我们可将上述四种算法的时间复杂度记为  $O(n^3)$ ， $O(n^2)$ ， $O(n)$  和  $O(1)$ 。时间复杂度不一定是全是多项式，还可能是  $O(2^n)$ ， $O(\log n)$ ， $O(n \log n)$  等等。

在着手编写之前先估算算法的复杂度，一些显然过不去的算法就没必要写了，一般把计算机的执行速度定为  $10^9$  次，我们可以列出算法的最大规模

运算量	$n$	$n!$	$2^n$	$n^2$	$n^3$	$n \log_2 n$
数据范围	$10^9$	13	29	31622	1000	$3.9 \times 10^7$
二倍速度	$2 \times 10^9$	13	30	44721	1259	$7.6 \times 10^7$

时间复杂度本身是很复杂的概念，有兴趣的可以查阅资料。因为时间很容易测不准，不能简单的以运行时间评判程序的优劣，尤其运行时间极其短时；同样也不能以上界分析的结果简单的断定程序的速度。在不少情况下，算法实际能解决的问题规模与上表有着较大差异。

尽管如此，此表还是有一定借鉴意义的，比如一个指明  $n \leq 20$  的题目可能  $2^n$  的算法已经足够，而  $n \leq 10^4$  的题目可能需要  $n^2$  的算法， $n \leq 10^6$  则可能至少要  $n \log n$  的算法了。

相对于充斥着各种奇妙优化的算法，朴素的算法常称为暴力。

我们应尽量思考，尝试发现更优的算法，以更佳的方法解决问题。

### 3 简单循环

大部分题还是常规的。

#### 3.1 7-1 输出等腰杨辉三角

样例有毒，仔细读题。应该没有人会真的写循环吧（  
连续的字符常量会自动合并。

---

```

1  int main() {
2      printf(
3          "          1\n"
4          "          1  1\n"
5          "          1  2  1\n"
6          "          1  3  3  1\n"
7          "          1  4  6  4  1\n"
8          "          1  5 10 10  5  1\n"
9      );
10     return 0;
11 }

```

---

### 3.2 7-2 含 8 的数字的个数

暴力即可。

---

```

1  int main() {
2      int a, b;
3      scanf("%d%d", &a, &b);
4      int sum = 0;
5      for (int i = a; i <= b; i++) {
6          int ti = i;
7          while (ti > 0) {
8              if (ti % 10 == 8) {
9                  sum++;
10                 break;
11             }
12             ti /= 10;
13         }
14     }
15     printf("%d", sum);
16     return 0;
17 }

```

---

假如要求的数字范围更大，设  $a, b$  为 `int` 范围的数据。

不妨令  $a = 1$ ,  $b = 123456987$  试试看你的算法要跑多久？怎么优化？详细见 [PTA 循环结构 H](#)。

### 3.3 7-3 立方和

把自身不断立方求和看作数字之间的链，那么关键在于判环。一种办法是记下所有的数，每增加一个数就和之前所有的数比对。有  $O(n^2)$  和  $O(n)$  两种写法。

但还有一种思考方式。注意到一条链的长度不可能超过 1000，而且要判的环是自身成环，于是暴力 1000 次即可。

整数的幂一般不要用 `pow`，因为其结果是 `double`，可能会有精度问题。

---

```
1 int sum(int x) {
2     int a = x % 10;
3     x /= 10;
4     int b = x % 10;
5     x /= 10;
6     int c = x % 10;
7     return a * a * a + b * b * b + c * c * c;
8 }
9
10 int main() {
11     int x;
12     scanf("%d", &x);
13     for (int i = 0; i < 1000; i++) {
14         x = sum(x);
15     }
16     if (sum(x) == x)
17         printf("%d\n", x);
18     else
19         printf("error");
20     return 0;
21 }
```

---

### 3.4 ! 7-4 统计单词数量

怎么题目没了？

没给数据，猜一手 1000086。

大数组不要开在函数里，尽量做为全局变量，否则数组过大可能会爆栈，会段溢出。

或者使用 `getchar` 做到在线运行，那样给多长都不怕了。

---

```
1 int isAlpha(int x) {
2     if ('A' <= x && x <= 'Z')
3         return 1;
4     else if ('a' <= x && x <= 'z')
5         return 1;
6     return 0;
7 }
8
9 char s[1000086];
10
11 int main() {
12     gets(s);
13     int len = strlen(s);
```

```

14     int p = isAlpha(s[0]);
15     int sum = 0;
16     for (int i = 1; i <= len; i++) {
17         int t = isAlpha(s[i]);
18         if (p && !t)
19             sum++;
20         p = t;
21     }
22     printf("%d", sum);
23     return 0;
24 }

```

---

### 3.5 7-6 中国余数定理

显然答案只会在  $1 \sim 105$  之间，暴力枚举即可，代码就不放了。  
中国剩余定理解法

```

1  int main() {
2      int a, b, c;
3      while (scanf("%d%d%d", &c, &a, &b) != EOF) {
4          int ans = a * 70 + b * 21 + c * 15;
5          ans = (ans - 1 + 105) % 105 + 1;
6          printf("%d\n", ans);
7      }
8      return 0;
9  }

```

---

解释见 [PTA 循环结构 H](#)

### 3.6 7-7 多项式求值

显然

$$\sum_{k=1}^n (-1)^{k+1} k^2 = (-1)^{n+1} \frac{n(n+1)}{2}$$

```

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

```

---

### 3.7 7-9 英文字母替换加密

---

```
1  int jiami(int x) {
2      if ('a' <= x && x <= 'z') {
3          x = (x - 'a' + 1) % 26 + 'A';
4      } else if ('A' <= x && x <= 'Z') {
5          x = (x - 'A' + 1) % 26 + 'a';
6      }
7      return x;
8  }
9
10 char s[100086];
11
12 int main() {
13     gets(s);
14     int len = strlen(s);
15     for (int i = 0; i < len; i++) {
16         s[i] = jiami(s[i]);
17     }
18     printf(s);
19     return 0;
20 }
```

---

### 3.8 7-10 字母塔

---

```
1  int main() {
2      int n;
3      scanf("%d", &n);
4      for (int i = 1; i <= n; i++) {
5          for (int j = 1; j <= n - i; j++)
6              printf(" ");
7          for (int j = 1; j <= i; j++)
8              printf("%c", j + 'A' - 1);
9          for (int j = i - 1; j >= 1; j--)
10             printf("%c", j + 'A' - 1);
11         printf("\n");
12     }
13     return 0;
14 }
```

---

### 3.9 7-12 质因子分解

试除即可。

---

```
1  int rst[1000], cnt;
2
3  int main() {
4      int n;
5      scanf("%d", &n);
6      if (n == 2) {
7          printf("2=2");
8          return 0;
9      }
10     printf("%d=", n);
11     while (n % 2 == 0) {
12         rst[++cnt] = 2;
13         n /= 2;
14     }
15     int sn = (int) sqrt(n * 1.0);
16     for (int i = 3; i <= sn; i += 2) {
17         while (n % i == 0) {
18             rst[++cnt] = i;
19             n /= i;
20         }
21     }
22     if (n > 1)
23         rst[++cnt] = n;
24     printf("%d", rst[1]);
25     for (int i = 2; i <= cnt; i++) {
26         printf(" *%d", rst[i]);
27     }
28     return 0;
29 }
```

---

### 3.10 7-13 启程几何

---

```
1  int main() {
2      int sa, sb;
3      scanf("%d %d", &sa, &sb);
4      int a = 5, b = 30, day = 1;
5      int sum = -sb;
6      while (sum < sa) {
7          day++;
8          a *= 2;
9          b += 30;
10         sum += a + b - sb;
```



```
11     }
12     printf("%d", day + 1);
13     return 0;
14 }
```

---

### 3.11 7-14 进度条

---

```
1  int main() {
2      int seed;
3      scanf("%d", &seed);
4      srand(seed);
5      for (int i = 1; i <= seed; i++) {
6          int t = 1 + rand() % 35;
7          for (int i = 1; i <= t; i++)
8              printf(">");
9          for (int i = t + 1; i <= 35; i++)
10             printf(".");
11         double d = t * 100 / 35.0;
12         printf(" %2.0lf%%\n", d);
13     }
14     return 0;
15 }
```

---

### 3.12 7-17 计算到任意日期的总天数

---

```
1  int a[13]={0,31,59,90,120,151,181,212,243,273,304,334,365};
2
3  int main() {
4      int y,m,d;
5      scanf("%d%d%d", &y, &m, &d);
6      int run = y / 4 - y/100 + y/400;
7
8      int sum = run + 365 * (y-1);
9      if(y % 400 == 0 || (y%4==0 && y%100!=0))
10         sum--;
11     printf("%d\n", sum);
12
13     sum += a[m-1];
14     if(y % 400 == 0 || (y%4==0 && y%100!=0))
15         sum++;
16     printf("%d\n", sum);
17 }
```

```
18     sum += d;
19     printf("%d\n", sum);
20     return 0;
21 }
```

---

### 3.13 7-18 最大值和最小值

---

```
1  int main() {
2      int n;
3      scanf("%d", &n);
4      int max = 0, min = 0x3f3f3f3f;
5      for (int i = 1; i <= n; i++) {
6          int t;
7          scanf("%d", &t);
8          max = max < t ? t : max;
9          min = min > t ? t : min;
10     }
11     printf("%d", max - min);
12     return 0;
13 }
```

---

### 3.14 7-19 计算评分

---

```
1  int main() {
2      int n;
3      scanf("%d", &n);
4      double sum = 0;
5      double min = 101, max = 0;
6      for (int i = 1; i <= n; i++) {
7          double t;
8          scanf("%lf", &t);
9          max = max < t ? t : max;
10         min = min > t ? t : min;
11         sum += t;
12     }
13     double score = (sum - min - max) / (n - 2);
14     printf("%.0lf %.1lf %.1lf", score, min, max);
15     return 0;
16 }
```

---

### 3.15 7-20 我们爱运动

---

```
1  double nn[30005];
2
3  int main() {
4      int n, a, b;
5      scanf("%d %d %d", &n, &a, &b);
6      for (int i = 1; i <= n; i++) {
7          scanf("%lf", &nn[i]);
8      }
9      int sa = 1, sb = 1;
10     for (int i = 1; i <= n; i++) {
11         if (nn[a] > nn[i])
12             sa++;
13         if (nn[b] > nn[i])
14             sb++;
15     }
16     printf("%d %d", sa, sb);
17     return 0;
18 }
```

---

### 3.16 7-21 累加 a-aa+aaa-aaaa+...

不难找规律。

---

```
1  int main() {
2      char a;
3      int n;
4      scanf("%c %d", &a, &n);
5      printf("sum=");
6      if(n % 2 == 0)
7          printf("-");
8      for (int i = 1; i <= n; i++) {
9          if(i % 2 == 1)
10             putchar(a);
11         else
12             putchar('0');
13     }
14     return 0;
15 }
```

---

### 3.17 7-23 循环 - n 个数最大值

---

```
1  int main() {
```

```

2     int n;
3     scanf("%d", &n);
4     if(n <= 0)
5         return 0;
6     int max = 0;
7     for (int i = 1; i <= n; i++) {
8         int t;
9         scanf("%d", &t);
10        max = max < t ? t : max;
11    }
12    printf("%d", max);
13    return 0;
14 }

```

---

### 3.18 7-25 求简单交错序列的前 N 项和

---

```

1  int main() {
2      int n;
3      scanf("%d", &n);
4      double sum = 0;
5      double flag = 1;
6      for (int i = 1; i <= n; i++) {
7          sum += flag * 1 / (2 * i - 1.0);
8          flag = -flag;
9      }
10     printf("sum = %.3lf\n", sum);
11     return 0;
12 }

```

---

### 3.19 7-26 谁最高

---

```

1  char s[100], cnt;
2
3  int main() {
4      double a, b, c;
5      scanf("%lf%lf%lf", &a, &b, &c);
6      double max = a > b ? a : b;
7      max = max > c ? max : c;
8      if (max - a <= 0.001) {
9          s[cnt] = ' ';
10         s[cnt + 1] = 'A';
11         cnt += 2;

```

```

12     }
13     if (max - b <= 0.001) {
14         s[cnt] = ' ';
15         s[cnt + 1] = 'B';
16         cnt += 2;
17     }
18     if (max - c <= 0.001) {
19         s[cnt] = ' ';
20         s[cnt + 1] = 'C';
21         cnt += 2;
22     }
23     s[cnt] = 0;
24     printf("%s", &s[1]);
25     return 0;
26 }

```

---

## 4 格式输出

格式输出挺烦的，要仔细。

建议先把答案存到数组里，再想办法套格式。

### 4.1 7-4 输出 2 到 n 之间的全部素数

暴力竟然能过。。正解应该是筛法。

---

```

1  int isprime(int n) {
2      if (n < 3)
3          return n == 2;
4      if (n % 2 == 0)
5          return 0;
6      // 否则每次循环都要开根
7      int sn = (int) sqrt(n * 1.0);
8      for (int i = 3; i <= sn; i += 2)
9          if (n % i == 0)
10             return 0;
11     return 1;
12 }
13
14 int cnt, prime[20000001];
15 void init(int n) {
16     for (int i = 2; i <= n; i++) {
17         if (isprime(i)) {
18             prime[++cnt] = i;

```

```

19     }
20 }
21 }
22
23 int main() {
24     int n;
25     scanf("%d", &n);
26     init(n);
27     for (int i = 1; i < cnt; i++) {
28         printf("%6d", prime[i]);
29         if (i % 10 == 0)
30             printf("\n");
31     }
32     printf("%6d", prime[cnt]);
33     return 0;
34 }

```

---

## 4.2 7-5 输出前 n 个 Fibonacci 数

---

```

1  int cnt, fib[20000001];
2  void init(int n) {
3      fib[1] = 1;
4      fib[2] = 1;
5      for (int i = 3; i <= n; i++)
6          fib[i] = fib[i - 1] + fib[i - 2];
7      cnt = n;
8  }
9
10 int main() {
11     int n;
12     scanf("%d", &n);
13     if (n < 1) {
14         printf("Invalid.");
15         return 0;
16     }
17     init(n);
18     for (int i = 1; i < cnt; i++) {
19         printf("%11d", fib[i]);
20         if (i % 5 == 0)
21             printf("\n");
22     }
23     printf("%11d\n", fib[cnt]);

```

```
24     return 0;
25 }
```

---

### 4.3 7-11 数字菱形

递归法

---

```
1  int n, h;
2
3  void printline(int l, int t) {
4      for (int i = 0; i < l; i++)
5          putchar(' ');
6      printf("%d", t);
7      if(l != h) {
8          for (int i = 1; i < (h - l) * 2; i++)
9              putchar(' ');
10         printf("%d", t);
11     }
12     printf("\n");
13 }
14
15 void solve(int x) {
16     printline(h - x, (n + x) % 10);
17     if(x == h)
18         return;
19     solve(x + 1);
20     printline(h - x, (n + x) % 10);
21 }
22
23 int main() {
24     scanf("%d %d", &n, &h);
25     h /= 2;
26     solve(0);
27     return 0;
28 }
```

---

循环法

---

```
1  #define INC(x) (x = (x + 1) % 10)
2  #define DEC(x) (x = (x - 1 + 10) % 10)
3
4  int main() {
5      int a, h;
6      scanf("%d %d", &a, &h);
```

```

7     int b = h / 2;
8     _fora(i, 1, b) printf(" ");
9     printf("%d\n", a);
10    INC(a);
11    _fora(i, 1, b) {
12        _fora(j, 1, b - i)
13            printf(" ");
14        printf("%d", a);
15        _fora(j, 1, i * 2 - 1)
16            printf(" ");
17        printf("%d", a);
18        INC(a);
19        printf("\n");
20    }
21    DEC(a);
22    DEC(a);
23    _forz(i, b - 1, 1) {
24        _fora(j, 1, b - i)
25            printf(" ");
26        printf("%d", a);
27        _fora(j, 1, i * 2 - 1)
28            printf(" ");
29        printf("%d", a);
30        DEC(a);
31        printf("\n");
32    }
33    _fora(i, 1, b)
34        printf(" ");
35    printf("%d\n", a);
36    return 0;
37 }

```

---

#### 4.4 7-15 重排矩阵

排序后再想办法绕圈圈。

---

```

1  int nn[10086];
2  int mtx[10086];
3  int tm, tn;
4
5  void quick_sort(int l, int r) {
6      if (l >= r)
7          return;

```



```

8     int i = l, j = r;
9     int x = nn[(l + r) / 2];
10    while (i <= j) {
11        while (nn[j] > x)
12            j--;
13        while (nn[i] < x)
14            i++;
15        if (i <= j) {
16            int t = nn[i];
17            nn[i] = nn[j];
18            nn[j] = t;
19            i++;
20            j--;
21        }
22    }
23    quick_sort(l, j);
24    quick_sort(i, r);
25 }
26
27 int min(int a, int b) {
28     return a > b ? b : a;
29 }
30
31 int pos(int x, int y) {
32     return (y - 1) * tn + x;
33 }
34
35 int main() {
36     int n = rr();
37     _fora(i, 1, n)
38         nn[i] = rr();
39     quick_sort(1, n);
40     int sn = (int)sqrt(n);
41     tn = 1;
42     _forz(i, sn, 1) {
43         if (n % i == 0) {
44             tn = i;
45             break;
46         }
47     }
48     tm = n / tn;
49     int p = n;

```

```

50     _fora(i, 1, (tn + 1) / 2) {
51         if (p <= 0)
52             break;
53         _fora(j, i, tn - i)
54             mtx[pos(j, i)] = nn[p--];
55         if (p <= 0)
56             break;
57         _fora(j, i, tm - i + 1)
58             mtx[pos(tn - i + 1, j)] = nn[p--];
59         if (p <= 0)
60             break;
61         _forz(j, tn - i, i)
62             mtx[pos(j, tm - i + 1)] = nn[p--];
63         if (p <= 0)
64             break;
65         _forz(j, tm - i, i + 1)
66             mtx[pos(i, j)] = nn[p--];
67     }
68     _fora(i, 1, tm) {
69         _fora(j, 1, tn - 1)
70             printf("%d ", mtx[pos(j, i)]);
71         printf("%d\n", mtx[pos(tn, i)]);
72     }
73     return 0;
74 }

```

---

## 4.5 7-16 1000 以内所有各位数字之和为 n 的正整数

---

```

1  int cnt, nn[20000001];
2  void init(int n) {
3      _fora(i, 1, 1000) {
4          int sum = 0;
5          int ti = i;
6          while (ti) {
7              sum += ti % 10;
8              ti /= 10;
9          }
10         if (sum == n)
11             nn[++cnt] = i;
12     }
13 }
14

```

```

15 int main() {
16     int n;
17     scanf("%d", &n);
18     init(n);
19     _fora(i, 1, cnt - 1) {
20         printf("%8d", nn[i]);
21         if (i % 6 == 0)
22             printf("\n");
23     }
24     printf("%8d\n", nn[cnt]);
25     return 0;
26 }

```

---

## 4.6 7-22 嵌套循环-素数的和

---

```

1 int notp[100000001];
2 int cnt, prime[200000001];
3 void init(int n) {
4     _fora(i, 2, n) {
5         if (!notp[i])
6             prime[++cnt] = i;
7         int t = n / i;
8         _fora(j, 1, cnt) {
9             if (prime[j] > t)
10                 break;
11             notp[i * prime[j]] = 1;
12             if (i % prime[j] == 0)
13                 break;
14         }
15     }
16 }
17
18 int main() {
19     init(1000000);
20     int a, b;
21     scanf("%d%d", &a, &b);
22     if (a > b) {
23         int t = a; a = b; b = t;
24     }
25     if (a < 0)
26         return 0;
27     int sum = 0;

```

```
28     _fora(i, a, b) {
29         if (!notp[i])
30             sum += i;
31     }
32     printf("%d", sum);
33     return 0;
34 }
```

---