

# 编译原理第三次实验测试用例：目录

<b>1</b>	<b>A 组测试用例</b>	<b>2</b>
1.1	A-1 . . . . .	2
1.2	A-2 . . . . .	2
1.3	A-3 . . . . .	3
1.4	A-4 . . . . .	4
1.5	A-5 . . . . .	5
<b>2</b>	<b>B 组测试用例</b>	<b>6</b>
2.1	B-1 . . . . .	6
2.2	B-2 . . . . .	7
2.3	B-3 . . . . .	9
<b>3</b>	<b>C 组测试用例</b>	<b>10</b>
3.1	C-1 . . . . .	11
3.2	C-2 . . . . .	13
<b>4</b>	<b>D 组测试用例</b>	<b>15</b>
4.1	D-1 . . . . .	15
<b>5</b>	<b>E 组测试用例</b>	<b>17</b>
5.1	E1-1 . . . . .	17
5.2	E1-2 . . . . .	18
5.3	E1-3 . . . . .	19
5.4	E2-1 . . . . .	21
5.5	E2-2 . . . . .	22
5.6	E2-3 . . . . .	23
<b>6</b>	<b>结束语</b>	<b>26</b>

## 1 A 组测试用例

本组测试用例共 5 个，均为比较简单的程序，简单检查针对赋值-算数语句、分支语句、循环语句、数组表达式和函数调用的翻译。

### 1.1 A-1

输入

```
1 int main() {  
2     int x, y, z;  
3     int a, b, c;  
4     int t = 3;  
5     x = 12;  
6     b = x * x + 13;  
7     y = b / 13 + 1;  
8     z = x / b + b * y;  
9     write(z);  
10    c = t + z * 2;  
11    write(c);  
12    t = x + b + y + x / c + t;  
13    write(t);  
14    return 0;  
15 }
```

程序输入：无；预期输出：2041 4085 185

说明：这个测试用例针对赋值与算术语句进行测试。注意，预期输入/输出中每个数字会占一行，这里为了节省空间写在同一行，以空格隔开（下同）。

### 1.2 A-2

输入

```
1 int main() {  
2     int dollar, rmb;  
3     dollar = read();  
4     rmb = read();
```

```

5      if (dollar > 100) {
6          if (rmb < 50) {
7              write(rmb + dollar);
8          } else {
9              write(rmb - dollar);
10         }
11     } else if (dollar == 100) {
12         if (rmb < 100) {
13             write(rmb);
14         } else {
15             write(rmb - 100);
16         }
17     } else if (dollar < 100) {
18         if (rmb + dollar > 100){
19             write(dollar + 100);
20         } else {
21             write(100);
22         }
23     }
24     write(dollar + rmb);
25     return 0;
26 }

```

输入：105 35；输出：140 140

输入：100 100；输出：0 200

输入：95 40；输出：195 135

输入：5 5；输入：100 10

说明：这个测试用例主要针对分支语句进行测试的小程序。注意，程序输入以空格隔开，每次输入一个数（下同）。

### 1.3 A-3

输入

```

1 int main() {

```

```

2      int x, n, result;
3      x = read();
4      n = read();
5      if (n == (n / 2 * 2)) {
6          result = 1;
7      } else {
8          result = x;
9      }
10     n = n / 2;
11     while (n > 0) {
12         x = x * x;
13         if (n != (n / 2 * 2)) {
14             result = result * x;
15         }
16         n = n / 2;
17     }
18     write(result);
19     return 0;
20 }

```

输入: 2 2; 输出: 4

输入: 3 3; 输出: 27

输入: 5 10; 输出: 9765625

输入: 1 25; 输出: 1

说明: 这个测试用例主要针对循环语句进行测试, 求 a 的 b 次幂。

## 1.4 A-4

输入

```

1  int main() {
2      int x[5], tem, i, j;
3      i = 0;
4      while(i < 5) {
5          x[i] = read();

```

```

6         i = i + 1;
7     }
8     i = 1;
9     while (i < 5) {
10         j = i;
11         while (j > 0 && x[j-1] > x[j]) {
12             tem = x[j];
13             x[j] = x[j-1];
14             x[j-1] = tem;
15             j = j - 1;
16         }
17         i = i + 1;
18     }
19     i = 0;
20     while (i < 5) {
21         write(x[i]);
22         i = i + 1;
23     }
24     return 0;
25 }

```

输入：32 20 15 19 12；输出：12 15 19 20 32

说明：这个测试用例主要针对一维数组进行测试，实现升序插入排序。

## 1.5 A-5

输入

```

1 int swap(int a, int b) {
2     int tem = a;
3     a = b;
4     b = tem;
5     write(a);
6     write(b);
7     return a;

```

```

8 }
9
10 int main() {
11     int x[5];
12     int i = 0;
13     while (i < 5) {
14         x[i] = read();
15         if (i > 0) {
16             swap(x[i-1], x[i]);
17         }
18         i = i + 1;
19     }
20     return 0;
21 }

```

输入：2 3 6 8 10； 输出：3 2 6 3 8 6 10 8

说明：这个测试用例主要针对函数的调用进行简单测试。

## 2 B 组测试用例

本组测试用例共 3 个，较 A 组测试用例复杂，这里不专门针对赋值和算术语句设计测试用例。

### 2.1 B-1

输入

```

1 int myPow(int x, int n) {
2     int value;
3     if (n == 0) { return 1; }
4     if (n == 1) { return x; }
5     if (n == 2) { return x * x; }
6     if (n == (n/2*2)) { return myPow(myPow(x, n/2), 2); }
7     else {
8         value = myPow(myPow(x, n/2), 2);

```

```

9         return x*value;}
10    }
11
12    int main() {
13        int x1, n1;
14        x1 = read();
15        n1 = read();
16        write(myPow(x1,n1));
17        return 0;
18    }

```

输入: 2 4; 输出: 16

输入: 11 5; 输出: 161051

输入: 12 3; 输出: 1728

输入: 15 0; 输出: 1

说明: 求 a 的 b 次幂的递归版本, 考察复杂的函数调用和递归。

## 2.2 B-2

输入

```

1    int countSort() {
2        int x[5], count[10], sorted[5], i;
3        i = 0;
4        while (i < 10) {
5            count[i] = 0;
6            i = i + 1;
7        }
8        i = 0;
9        while (i < 5) {
10           x[i] = read();
11           count[x[i]] = count[x[i]] + 1;
12           i = i + 1;
13        }
14        i = 1;

```

```

15     while ( i < 10) {
16         count[i] = count[i] + count[i-1];
17         i = i + 1;
18     }
19     i = 0;
20     while ( i < 5 ) {
21         sorted[count[x[i]]-1] = x[i];
22         count[x[i]] = count[x[i]] - 1;
23         i = i + 1;
24     }
25     i = 0;
26     while (i < 5) {
27         write(sorted[i]);
28         i = i + 1;
29     }
30     return 0;
31 }
32
33 int bubbleSort(){
34     int a[5], b, c, tem;
35     b = 0;
36     while (b < 5) {
37         a[b] = read();
38         b = b + 1;
39     }
40     c = 1;
41     while (c == 1) {
42         c = 0;
43         b = 1;
44         while (b < 5) {
45             if (a[b] < a[b-1]) {
46                 c = 1;

```



```

47         tem = a[b-1];
48         a[b-1] = a[b];
49         a[b] = tem;
50     }
51     b = b + 1;
52 }
53 }
54 b = 0;
55 while (b < 5) {
56     write(a[b]);
57     b = b + 1;
58 }
59 return 0;
60 }
61
62 int main(){
63     countSort();
64     bubbleSort();
65     return 0;
66 }

```

输入：5 3 2 4 1 5 3 2 4 1；输出：1 2 3 4 5 1 2 3 4 5

说明：实现计数排序和冒泡排序的数组排序程序。

## 2.3 B-3

输入

```

1 int search(int target) {
2     int x[5], left, right, index, middle;
3     int i = 0;
4     while(i < 5) {
5         x[i] = read();
6         i = i + 1;
7     }

```

```

8      left = 0;
9      right = 4;
10     while (left <= right) {
11         index = (left + right) / 2;
12         middle = x[index];
13         if (middle == target) {return index;}
14         if ((middle > x[left] && target >= x[left] && target
15             < middle) || (middle < x[left] && (target >= x[
16                 left] || target < middle))) {
17             right = index - 1;
18         } else {
19             left = index + 1;
20         }
21     }
22     return -1;
23 }
24
25 int main() {
26     int n;
27     n = read();
28     write(search(n));
29     return 0;
30 }

```

输入: 0 6 7 0 1 2; 输出: 2

输入: 3 6 7 6 8 0 2 4; 输出: -1

说明: 谷歌笔试题, 一个升序有序数组 (无重复元素, 如输入 1 中 0 1 2 6 7) 以未知位置为中心对调 (6 7 0 1 2), 二分搜索特定元素 (0), 返回其数组中位置 (2) 或 -1 (若不存在)。

### 3 C 组测试用例

本组测试用例共 2 个, 是经典问题。

### 3.1 C-1

输入

```
1 int main(){
2     int count, i, valid, j, k, cont, n;
3     int place[8];
4     int row[8];
5     int ldiag[8];
6     int rdiag[8];
7     count = 0;
8     i = 0;
9     n = read();
10    while(i < n) {
11        place[i] = -1;
12        i = i + 1;
13    }
14    i = 0;
15    cont = 1;
16    while(cont == 1) {
17        if (i == n) {
18            valid = 1;
19            j = 0;
20            while (j < n) {
21                row[j] = 1;
22                ldiag[j] = 1;
23                rdiag[j] = 1;
24                j = j + 1;
25            }
26            j = 0;
27            while (j < n) {
28                if(row[place[j]] != 1
29                || ldiag[place[j]] != 1
30                || rdiag[place[j]] != 1) {
```

```

31         valid = 0;
32         j = n;

33     } else {
34         row[place[j]] = 0;
35         k = 0;
36         while (k < n-1) {
37             ldiag[k]
38                 = ldiag[k + 1];
39             k = k + 1;

40         }
41         ldiag[n-1] = 1;
42         if (place[j] != 0)
43             { ldiag[place[j] - 1] = 0;}
44         k = n-1;
45         while(k > 0) {
46             rdiag[k]
47                 = rdiag[k - 1];
48             k = k - 1;

49         }
50         rdiag[0] = 1;
51         if (place[j] != n-1)
52             {rdiag[place[j] + 1] = 0;}
53         j = j + 1;
54     }
55 }
56 if (valid == 1) {
57     count = count + 1;

```

```

58         }
59         i = i - 1;
60     } else {
61         while (i >= 0 && place[i] >= n-1) {
62             place[i] = -1;
63             i = i - 1;
64         }
65         if (i == -1) {
66             cont = 0;
67         } else {
68             place[i] = place[i] + 1;
69             i = i + 1;
70         }
71     }
72 }
73 write(count);
74 return 0;
75 }

```

输入：2；输出：0 输入：4；输出：2

说明：这个测试用例实现了经典 N 皇后问题，输入 N，输出所有可能的摆放方式数目。

### 3.2 C-2

输入

```

1 int trap() {
2     int lh = 0, lIndex = 0, i = 0, count = 0, thisPool = 0;
3     int n = 12;
4     int rh = 0;
5     int height[12];
6     while (i < n) {
7         height[i] = read();
8         i = i + 1;

```

```

9      }
10     i = 0;
11     while (i < n && height[i] == 0) {
12         i = i + 1;
13     }
14     if (i >= n) {
15         return 0;
16     }
17     lh = height[i];
18     lIndex = i;
19     thisPool = 0;
20     i = i + 1;
21     while (i < n) {
22         if (height[i] < lh){
23             thisPool = thisPool + (lh - height[i]);
24         }else {
25             count = count + thisPool;
26             lh = height[i];
27             lIndex = i;
28             thisPool = 0;
29         }
30         i = i + 1;
31     }
32     thisPool = 0;
33     rh = 0;
34     i = n - 1;
35     while (i > lIndex && height[i] == 0) {
36         i = i - 1;
37     }
38     rh = height[i];
39     i = i - 1;
40     while (i > lIndex) {

```

```

41         if (height[i] < rh) {
42             thisPool = thisPool + (rh - height[i]);
43         } else {
44             count = count+ thisPool;
45             rh = height[i];
46             thisPool = 0;
47         }
48         i = i - 1;
49     }
50     return count + thisPool;
51 }
52
53 int main () {
54     int result = trap();
55     write(result);
56     return 0;
57 }

```

输入：0 1 0 2 1 0 1 3 2 1 2 1；输出：6

说明：给出一个  $n$  维的非负数组代表一个海拔图，每个数字代表的宽度为 1，计算下雨后一共可以积存多少水。

## 4 D 组测试用例

本组测试用例共 1 个，主要用于测试中间代码的优化。

### 4.1 D-1

输入

```

1 int process(int x) {
2     int y = 3;
3     y = 11 * 3 - 2 + 5;
4     y = x * 321 * 2 + x * y - x + y * x + y * y + x + x - 23 +
        45;

```

```

5      y = y / 3 + 14 * 24 - x * 12 / 4 - 20 * 3 + y / 12 * 24 + 12
      * 3 + 3 / 2;
6      y = x + 4 * 6 + 3 / 2;
7      return y;
8  }
9
10 int main () {
11     int a = 5 / 2 + 14 - 3, b = 7 * 5 / 2 + 3, c = 4 + 5 + 6 - 1 /
        2;
12     int d = a + b + c;
13     int e = a * b + c / 2;
14     int f = a - b - c;
15     int g1 = 42, i = 0;
16     int g, h;
17     f = a + b + c + 1000 * 2 - f;
18     while (a + b < f) {
19         g1 = g1 + i * 12 + 4 + 5 + 7 / 3;
20         g = process(f) + 2 * a - f + c * d;
21         i = i + i;
22         i = i + i;
23         i = i + i;
24         i = i + i;
25         i = i + i;
26         h = i + 3;
27         h = h - 1;
28         h = h + 3;
29         h = h - 3 * 2;
30         if (process(a) == process(a + 3 - 2 - 1)) {
31             f = f - 2 + 1;
32         }
33         a = a + 2 + 1;
34     }

```



```

35     h = g1 - 3 * 4;
36     while (h < g1) {
37         f = 15 * 4 - 2 + a;
38         g = g1 - 12;
39         h = h + 1;
40         g = g1;
41         i = a + b;
42         c = a + b;
43     }
44     write(f);
45     a = a + b;
46     b = a + b;
47     c = a + b;
48     f = a + b;
49     g = a + b;
50     write(c+f+g);
51     return 0;
52
53 }

```

输入：无；输出：1601 9438

说明：程序中有多个可优化点，包括常量折叠，公共子表达式等。首先需要保证中间代码的正确性，要能准确输出最后的结果，才能参加后面的效率竞赛。

## 5 E 组测试用例

本组测试用例共 6 个，针对不同分组进行测试。

E1 组针对 3.1 分组测试结构体的翻译，E2 组针对 3.2 分组测试一维数组作为参数和高维数组的翻译。每组 3 个测试用例。

### 5.1 E1-1

输入

```

1 struct Product{

```

```

2         int type;
3         int name;
4     };
5
6 int main () {
7     struct Product cola;
8     cola.type = read();
9     cola.name = 3;
10    write(cola.type+cola.name);
11    return 0;
12 }

```

输入：3；输出：6

说明：测试对于简单结构体的翻译，不涉及与数组的交互和结构体作为函数参数调用。针对 3.1 分组，其他分组同学需要提示无法翻译且不输出中间代码。

## 5.2 E1-2

输入

```

1 struct Product{
2     int type;
3     int name;
4 };
5
6 int main(){
7     struct Product cola[10];
8     int i, j, add, N = 10;
9     i = 0;
10    while(i < N) {
11        cola[i].type = 10;
12        cola[i].name = i;
13        i = i + 1;
14    }
15    i = 0;

```

```

16     add = 0;
17     while (i < N) {
18         j = 0;
19         add = add + cola[i].name;
20         while (j < N) {
21             cola[i].type = cola[i].type + add * cola[j].
22                 name;
23             j = j + 1;
24         }
25         i = i + 1;
26     }
27     write(cola[N-1].type);
28     return 0;
29 }

```

输入：无；输出：2035

说明：测试对于结构体作为数组的类型。针对 3.1 分组，其他分组同学需要提示无法翻译且不输出中间代码。

### 5.3 E1-3

输入

```

1 struct Student{
2     int name;
3     int grade;
4 };
5
6 struct Class{
7     struct Student students[50];
8     int average;
9 };
10
11 int calculate(struct Class class) {
12     int sum = 0, i = 0, N = 50;

```

```

13     while (i < N) {
14         sum = sum + class.students[i].grade;
15         i = i + 1;
16     }
17     class.average = sum / N;
18     return sum / N;
19 }
20
21 int main() {
22     struct Class school[10];
23     int i1 = 0, j1 = 0, N1 = 50, N2 = 10;
24     while (i1 < N2) {
25         j1 = 0;
26         while (j1 < N1) {
27             school[i1].students[j1].grade = i1 + j1 * 5;
28             j1 = j1 + 1;
29         }
30         i1 = i1 + 1;
31     }
32     j1 = 0;
33     i1 = 0;
34     while (i1 < N2) {
35         j1 = j1 + calculate(school[i1]);
36         i1 = i1 + 1;
37     }
38     write(j1);
39     return 0;
40 }

```

输入：无；输出：1265

说明：测试对于较复杂的结构体及其作为函数参数进行函数的调用。针对 3.1 分组，其他分组同学需要提示无法翻译且不输出中间代码。

## 5.4 E2-1

输入

```
1 int main() {
2     int dis[10][5][3];
3     int d1 = 10, d2 = 5, d3 = 3;
4     int i = 0, j = 0, k = 0, sum = 0;
5     while (i < d1) {
6         j = 0;
7         while (j < d2) {
8             k = 0;
9             while (k < d3) {
10                 dis[i][j][k] = i * d1 + j * d2 + d3 /
11                     (k+1);
12                 k = k + 1;
13             }
14             j = j + 1;
15         }
16         i = i + 1;
17     }
18     i = 0;
19     j = 0;
20     k = 0;
21     while (i < d1) {
22         j = 0;
23         while (j < d2) {
24             k = 0;
25             while (k < d3) {
26                 if (dis[i][j][k] > dis[0][0][0]) {
27                     sum = sum + dis[i][j][k];
28                 }
29                 k = k + 1;
30             }
31             j = j + 1;
32         }
33         i = i + 1;
34     }
35 }
```

```

29         }
30         j = j + 1;
31     }
32     i = i + 1;
33 }
34 write(sum);
35 return 0;
36 }

```

输入：无；输出：8495

说明：测试对于简单高维数组的翻译，不涉及数组作为函数参数。针对 3.2 分组，其他分组同学需要提示无法翻译且不输出中间代码。

## 5.5 E2-2

输入

```

1 int qsort(int array[10], int l, int r) {
2     int x = array[l], a = l, b = r;
3     if (a < b) {
4         while (a < b) {
5             while (a < b && array[b] > x) {b = b - 1;}
6             if (a < b) {array[a] = array[b]; a = a + 1;}
7             while (a < b && array[a] < x) {a = a + 1;}
8             if (a < b) {array[b] = array[a]; b = b - 1;}
9         }
10        array[a] = x;
11        qsort(array,l,a-1);
12        qsort(array,a+1,r);
13    }
14    return 0;
15 }
16
17 int main() {
18     int number[10], N = 10, i = 0;

```

```

19     while (i < N) {
20         number[i] = read();
21         i = i + 1;
22     }
23     qsort(number, 0, N-1);
24     i = 0;
25     while (i < N) {
26         write(number[i]);
27         i = i + 1;
28     }
29     return 0;
30 }

```

输入：8 9 6 5 4 7 1 2 3 0； 输出：0 1 2 3 4 5 6 7 8 9

说明：测试对于数组作为函数参数的翻译，实现快速排序。针对 3.2 分组，其他分组同学需要提示无法翻译且不输出中间代码。

## 5.6 E2-3

输入

```

1 int display(int chess[10], int number[1], int sum){
2     int board[10][10], i1 = 0, j1 = 0, tem = 1;
3     if (number[0] == 1) {
4         while (i1 < sum) {
5             j1 = 0;
6             tem = 1;
7             while (j1 < sum) {
8                 if (j1 == chess[i1]) {
9                     board[i1][j1] = 1;
10                    tem = tem * 10 + 1;
11                } else {
12                    board[i1][j1] = 0;
13                    tem = tem * 10;
14                }

```

```

15         j1 = j1 + 1;
16     }
17     write(tem);
18     i1 = i1 + 1;
19 }
20 }
21 return 0;
22 }
23
24 int dfs(int p[10], int r[10], int ld[10], int rd[10], int current,
25         int target, int c[1]){
26     int j = 0, nld[10], nrd[10], k;
27     if (current == target) {
28         c[0] = c[0] + 1;
29         display(p,c,target);
30         return 0;
31     }
32     while (j < target) {
33         if (r[j] == 1 && ld[j] == 1 && rd[j] == 1 ) {
34             p[current] = j;
35             r[j] = 0;
36             k = 0;
37             while (k< target - 1){
38                 nld[k] = ld[k + 1];
39                 k = k + 1;
40             }
41             nld[target -1] = 1;
42             if (j != 0) {
43                 nld[j - 1] = 0;
44             }
45             k = target-1;
46             while (k > 0){

```



```

46         nrd[k] = rd[k-1];
47         k = k - 1;
48     }
49     nrd[0] = 1;
50     if (j != target - 1){
51         nrd[j + 1] = 0;
52     }
53     dfs(p, r, nld, nrd, current + 1, target, c);
54     r[j] = 1;
55 }
56 j = j + 1;
57 }
58 return 0;
59 }
60 int main() {
61     int place[10], N, count[1];
62     int row[10], ldiag[10], rdiag[10] ,i = 0;
63     N = read();
64     if (N == 0 || N > 10) { return 0;}
65     while(i < N) {
66         row[i] = 1;
67         ldiag[i] = 1;
68         rdiag[i] = 1;
69         i = i + 1;
70     }
71     count[0] = 0;
72     dfs(place, row, ldiag, rdiag, 0, N, count);
73     write(count[0]);
74     return 0;
75 }

```

输入: 8; 输出: 110000000 100001000 100000001 100000100 100100000 100000010 101000000  
100010000 92

说明：测试对于较复杂的数组操作的翻译，是一个八皇后问题，输出第一个搜索到的摆放方案（每行 1 开头，之后八位代表摆放，1 代表放置皇后），并输出总共的解法数目。针对 3.2 分组，其他分组同学需要提示无法翻译且不输出中间代码。

## 6 结束语

如果对本测试用例有任何疑议，可以写邮件与王珏助教联系，注意同时抄送给许老师，本学期编译原理实验到此结束，祝愿大家都能取得好的成绩。