

编译原理第三次实验测试用例：目录

1	A 组测试用例	2
1.1	A-1	2
1.2	A-2	2
1.3	A-3	3
1.4	A-4	5
1.5	A-5	6
2	B 组测试用例	8
2.1	B-1	8
2.2	B-2	9
2.3	B-3	11
3	C 组测试用例	14
3.1	C-1	14
3.2	C-2	16
4	D 组测试用例	17
4.1	D-1	17
5	E 组测试用例	20
5.1	E1-1	20
5.2	E1-2	21
5.3	E1-3	22
5.4	E2-1	24
5.5	E2-2	25
5.6	E2-3	26
6	结束语	28

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/算术语句、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 result = 0;
3     int i = 9, j = 32, k = 57;
4     i = i * (i * j - k);
5     write(i);
6     j = i - (j + k * (k - 1)) * (-1);
7     write(j);
8     k = j - (k + k * j + 1 * (i + j + k)) / (i / 2);
9     write(k);
10    result = i + j - k;
11    write(result);
12    return 0;
13 }
```

程序输入: 无; 预期输出: 2079 5303 5005 2377

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 x;
3     int y;
4     int z;
5     x = read();
6     y = read();
```

```

7   z = read();
8   if (x > 0 && y > 0) {
9       write(1);
10  } else {
11      if (x == 0 && y == 0) {
12          write(2);
13      } else {
14          if (x < 0 && y < 0) {
15              write(3);
16          } else {
17              if (x >= y) {
18                  write(4);
19              } else {
20                  if (x < z && z < y) {
21                      write(5);
22                  } else {
23                      write(6);
24                  }
25              }
26          }
27      }
28  }
29  return 0;
30 }

```

程序输入: 1 2 3; 预期输出: 1

程序输入: 0 0 2; 预期输出: 2

程序输入: -3 2 1; 预期输出: 5

程序输入: -3 4 5; 预期输出: 6

说明: 主要针对分支语句进行测试。

1.3 A-3

输入

```

1  int main() {
2      int i;
3      int j;
4      int o_cnt = 0;
5      int i_cnt = 0;
6      int len = 10;
7      int arr[10];
8      int result = 0;
9
10     i_cnt = 0;
11     while (i_cnt < len) {
12         arr[i_cnt] = 0;
13         i_cnt = i_cnt + 1;
14     }
15
16     o_cnt = 0;
17     while (o_cnt < 3) {
18         i = read();
19         j = read();
20         if (i > 10) {
21             i = 1 + i - i / 10 * 10;
22         }
23         if (j > 10) {
24             j = 2 + j - j / 10 * 10;
25         }
26
27         i_cnt = 0;
28         while (i_cnt < len) {
29             arr[i_cnt] = arr[i_cnt] / 2 + i - j;
30             i_cnt = i_cnt + 1;
31         }
32         o_cnt = o_cnt + 1;

```

```

33     }
34
35     i_cnt = 0;
36     while (i_cnt < len) {
37         result = result + arr[i_cnt];
38         i_cnt = i_cnt + 1;
39     }
40     write(result);
41     return 0;
42 }

```

程序输入: 1 2 3 4 5 6; 预期输出: -20

程序输入: 1 3 5 7 9 11; 预期输出: 40

程序输入: -1 2 -3 4 -5 6; 预期输出: -160

程序输入: 1 1 2 3 5 8; 预期输出: -40

说明: 主要测试循环语句。

1.4 A-4

输入

```

1  int main() {
2      int i;
3      int j;
4      int cnt;
5      int len = 10;
6      int arr1[10];
7      int arr2[10];
8      int result = 0;
9
10     i = read();
11     cnt = 0;
12     while (cnt < len) {
13         arr1[cnt] = i + cnt;
14         cnt = cnt + 1;

```

```

15     }
16
17     arr2[0] = 2;
18     cnt = 1;
19     while (cnt < len) {
20         int tmp = arr1[cnt] - arr1[cnt] / 3 * 3;
21         if (tmp == 0) {
22             arr2[cnt] = arr1[cnt];
23         } else {
24             arr2[cnt] = - arr1[cnt];
25         }
26         cnt = cnt + 1;
27     }
28
29     cnt = 0;
30     while (cnt < len) {
31         result = result + arr2[cnt];
32         cnt = cnt + 1;
33     }
34     write(result);
35     return 0;
36 }

```

程序输入: 10; 预期输出: -43

程序输入: 100; 预期输出: -313

程序输入: 45; 预期输出: -142

程序输入: 90; 预期输出: -277

说明: 主要测试一维数组。

1.5 A-5

输入

```

1 int cal_sum(int s1, int s2, int s3) {
2     int s_res = s1 + s2 + s3;

```

```

3     return s_res;
4 }
5
6 int mod(int m1, int m2) {
7     int m_res = m1 - m1 / m2 * m2;
8     return m_res;
9 }
10
11 int is_good(int x) {
12     int a = x * 2;
13     int b = x + 2;
14     int c = x * x;
15     int sum = cal_sum(a, b, c);
16     int rem = mod(sum, 5);
17     if (rem > 1) {
18         return 1;
19     } else {
20         return 0;
21     }
22 }
23
24 int main() {
25     int f;
26     int g;
27     int h;
28     f = read();
29     g = read();
30     h = read();
31     if (is_good(f)) {
32         write(cal_sum(f, g, h));
33     } else {
34         write(0);

```

```
35     }
36     return 0;
37 }
```

程序输入: 2 2 2; 预期输出: 6

程序输入: 1 2 3; 预期输出: 0

说明: 一个测试函数调用的小程序。

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 mod(int m1, int m2) {
2     int m_res = m1 - m1 / m2 * m2;
3     return m_res;
4 }
5
6 int is_palindrome_number(int x) {
7     int div = 1;
8     if (x < 0) {
9         return 0;
10    }
11
12    while ((x / div) >= 10) {
13        div = div * 10;
14    }
15
16    while (x) {
17        if ((x / div) != mod(x, 10)) {
18            return 0;
```



```

19     }
20     x = (mod(x, div)) / 10;
21     div = div / 100;
22 }
23 return 1;
24 }
25
26 int main() {
27     int i;
28     i = read();
29     if (is_palindrome_number(i)) {
30         write(1);
31     } else {
32         write(0);
33     }
34     return 0;
35 }

```

程序输入: 1888988881; 预期输出: 0

程序输入: 13344331; 预期输出: 1

说明: 一个判断回文数的程序。

2.2 B-2

输入

```

1 int main() {
2     int n1;
3     int n2;
4     int a1[10];
5     int a2[10];
6     int a[20];
7     int i = 0, j = 0, k = 0, n = 0;
8     int cnt = 0;
9

```

```

10  n1 = read();
11  if (n1 > 10) {
12      n1 = 10;
13  }
14  while (cnt < n1) {
15      a1[cnt] = read();
16      cnt = cnt + 1;
17  }
18
19  n2 = read();
20  if (n2 > 10) {
21      n2 = 10;
22  }
23  cnt = 0;
24  while (cnt < n2) {
25      a2[cnt] = read();
26      cnt = cnt + 1;
27  }
28
29  while (i < n1 && j < n2) {
30      if (a1[i] < a2[j]) {
31          a[k] = a1[i];
32          k = k + 1;
33          i = i + 1;
34      } else {
35          a[k] = a2[j];
36          k = k + 1;
37          j = j + 1;
38      }
39  }
40
41  while (i < n1) {

```

```

42     a[k] = a1[i];
43     k = k + 1;
44     i = i + 1;
45 }
46 while (j < n2) {
47     a[k] = a2[j];
48     k = k + 1;
49     j = j + 1;
50 }
51
52 if ((n1 + n2 == 0) || (n1 + n2 == 1)) {
53     write(a[n1 + n2 - 1]);
54 } else if (n1 + n2 == 2) {
55     write((a[n1 + n2 - 1] + a[n1 + n2 - 2]) / 2);
56 } else {
57     n = n1 + n2;
58     if ((n - (n / 2) * 2) == 0) {
59         write((a[n / 2] + a[n / 2 - 1]) / 2);
60     } else {
61         write(a[n / 2]);
62     }
63 }
64 return 0;
65 }

```

程序输入: 3 1 4 5 4 4 8 9 10 预期输出: 5

程序输入: 5 -1 2 4 29 100 3 -10 3 10 预期输出: 3

说明: 一个计算两个排序好的数组合并后的中位数的程序。

2.3 B-3

输入

```

1 int main() {
2     int idx, s, e;

```

```

3  int tmp1, tmp2;
4  int len = 5;
5  int a[5];
6  int cnt = 0;
7  int stop = 0;
8  while (cnt < len) {
9      a[cnt] = read();
10     cnt = cnt + 1;
11 }
12
13 cnt = len / 2 - 1;
14 while (cnt >= 0) {
15     s = cnt;
16     e = len;
17     tmp1 = a[s];
18     stop = 0;
19     while ((s * 2 + 1 < e) && !stop) {
20         idx = s * 2 + 1;
21         if ((idx + 1 < e) && a[idx + 1] > a[idx]) {
22             idx = idx + 1;
23         }
24         if (a[idx] > tmp1) {
25             a[s] = a[idx];
26             s = idx;
27         } else {
28             stop = 1;
29         }
30     }
31     a[s] = tmp1;
32     cnt = cnt - 1;
33 }
34

```

```

35     cnt = len - 1;
36     while (cnt >= 0) {
37         tmp2 = a[0];
38         a[0] = a[cnt];
39         a[cnt] = tmp2;
40         s = 0;
41         e = cnt;
42         tmp1 = a[s];
43         stop = 0;
44         while ((s * 2 + 1 < e) && !stop) {
45             idx = s * 2 + 1;
46             if ((idx + 1 < e) && a[idx + 1] > a[idx]) {
47                 idx = idx + 1;
48             }
49             if (a[idx] > tmp1) {
50                 a[s] = a[idx];
51                 s = idx;
52             } else {
53                 stop = 1;
54             }
55         }
56         a[s] = tmp1;
57         cnt = cnt - 1;
58     }
59
60     cnt = 0;
61     while (cnt < len) {
62         write(a[cnt]);
63         cnt = cnt + 1;
64     }
65     return 0;
66 }

```

程序输入: 5 4 3 2 1 预期输出: 1 2 3 4 5

程序输入: 10 -3 29 100 2 预期输出: -3 2 10 29 100

说明: 堆排序。

3 C 组测试用例

本组测试用例共 2 个, 是较经典的问题。

3.1 C-1

输入

```
1 int cal_mid(int c1, int c2) {
2     return (c1 + c2) / 2;
3 }
4
5 int main() {
6     int len = 5;
7     int a[5];
8     int cnt = 0;
9     int i = 0, j = 0, tmp = 0;
10    int low, high;
11    int key;
12    while (cnt < len) {
13        a[cnt] = read();
14        cnt = cnt + 1;
15    }
16
17    i = 0;
18    while (i < len - 1) {
19        j = 0;
20        while (j < len - 1) {
21            if (a[j] > a[j + 1]) {
22                tmp = a[j];
23                a[j] = a[j + 1];
```

```

24         a[j + 1] = tmp;
25     }
26     j = j + 1;
27 }
28 i = i + 1;
29 }
30
31 i = 0;
32 while (i < len) {
33     write(a[i]);
34     i = i + 1;
35 }
36
37 key = read();
38
39 low = 0;
40 high = len - 1;
41 while (low <= high) {
42     int mid = cal_mid(low, high);
43     if (a[mid] == key) {
44         write(1);
45         return 0;
46     } else if (a[mid] < key) {
47         low = mid + 1;
48     } else {
49         high = mid - 1;
50     }
51 }
52 write(0);
53 return 0;
54 }

```

程序输入: 1 100 20 10 2 10 预期输出: 1 2 10 20 100 1

程序输入: 7 20 -1 23 3 4 -2 预期输出: -1 3 7 20 23 0

说明: 冒泡排序和二分查找。

3.2 C-2

输入

```
1 int mod(int m1, int m2) {
2     int m_res = m1 - m1 / m2 * m2;
3     return m_res;
4 }
5
6 int is_prime(int n) {
7     int i = 0;
8     if (n < 2) {
9         return 0;
10    }
11    if (n == 2) {
12        return 1;
13    }
14    if (mod(n, 2) == 0) {
15        return 0;
16    }
17
18    i = 3;
19    while ((i * i) <= n) {
20        if (mod(n, i) == 0) {
21            return 0;
22        }
23        i = i + 1;
24    }
25    return 1;
26 }
27
```



```

28 int fabonacci(int f) {
29     if (f == 0) {
30         return 0;
31     } else if (f == 1) {
32         return 1;
33     }
34     return fabonacci(f - 1) + fabonacci(f - 2);
35 }
36
37 int main() {
38     int idx = 0;
39     while (idx < fabonacci(5)) {
40         if(is_prime(idx)) {
41             write(idx);
42         }
43         idx = idx + 1;
44     }
45     return 0;
46 }

```

程序输入: 无; 预期输出: 2 3

说明: fabonacci 数列和素数判断。

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 mod(int a, int b) {
2     return a - (a / b) * b;
3 }
4

```

```

5  int do_work(int n) {
6      int x1, x2, x3;
7      n = mod(n, 5);
8      x1 = 2 * n;
9      x2 = n + n;
10     x3 = 4 * 4;
11     x1 = x3 + x2 + x1;
12     x2 = x1 + (x1 + 1) * (x2 + 1) * (x3 + 1);
13     x3 = x1 * (x1 + 1) + (x2 + 1) * (x2 + 1) + (x3 + 1) * (x3 + 1);
14     x1 = (x1 + 1) + (x2 + 1) + (x3 + 1);
15     return x1 + x2 + x3;
16 }
17
18 int main() {
19     int t1 = 2;
20     int t2 = 5;
21     int t3 = 10;
22     int i = 3 * (t2 * t3) - 100 / 5;
23     int j = 42 - (t1 * t2) * (t1 * (t2 * t3) / 32) + 100;
24     int k = 3 * 4 * 5 - 10 - (-(t1 * t2) * 3) - (t1 * t2) - (t1 * t2) +
        3 + 2 + 1;
25     int cnt = 0;
26     int sum = 0;
27     int len = t1 * t2;
28     int array[10];
29     int brray[10];
30     while (cnt < k) {
31         sum = sum + 1;
32         array[mod(cnt, len)] = i;
33         i = i + 1;
34         cnt = cnt + 1;
35     }

```

```

36  cnt = 0;
37  while (cnt < 100) {
38      brray[mod(cnt, len)] = i;
39      i = i + 1;
40      cnt = cnt + 1;
41  }
42
43  cnt = 0;
44  while (cnt < mod(do_work(10), 30)) {
45      j = array[mod(cnt, len)];
46      sum = sum + (-1) * cnt + do_work(j);
47      j = j + 2 * cnt;
48      j = j + 2 * cnt;
49      j = j + 2 * cnt;
50      if (mod(do_work(10), 10) == mod(do_work(j), 10)) {
51          j = mod(j, 10);
52      } else {
53          j = mod(j, 20);
54      }
55      array[mod(cnt, len)] = j * j;
56      cnt = cnt + 1;
57  }
58  j = j + array[0] + array[1];
59  j = j + array[0] + array[1];
60  j = j + array[0] + array[1];
61  j = j + array[0] + array[1];
62  write(j);
63  write(array[0]);
64  write(array[1]);
65  write(array[2]);
66  write(array[3]);
67  write(array[4]);

```

```
68     return 0;
69 }
```

程序输入: 无; 预期输出: 500 0 121 256 361 64

说明: 用于效率测试。

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 Animal {
2     int weight;
3     int height;
4     int index;
5 };
6
7 int main() {
8     struct Animal a, b;
9     a.weight = 20;
10    a.height = 30;
11    b.weight = 25;
12    b.height = 15;
13    a.index = a.weight * a.weight / (a.height * a.height);
14    b.index = b.weight * b.weight / (b.height * b.height);
15    write(a.index + b.index);
16    return 0;
17 }
```

程序输入: 无; 预期输出: 2

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 Point {
2     int x;
3     int y;
4     int z;
5 };
6
7 int main() {
8     int cnt = 0;
9     int sum = 0;
10    int len = 10;
11    struct Point points[10];
12
13    while (cnt < len) {
14        points[cnt].x = cnt;
15        points[cnt].y = cnt + cnt;
16        points[cnt].z = cnt * cnt;
17        cnt = cnt + 1;
18    }
19
20    cnt = 0;
21    while (cnt < len) {
22        if (cnt < 5) {
23            sum = sum + points[cnt].y + points[cnt].z;
24        } else {
25            sum = sum + points[cnt].x + points[cnt].y;
26        }
27        cnt = cnt + 1;
```

```
28     }
29     write(sum);
30     return 0;
31 }
```

程序输入: 无; 预期输出: 155

说明: 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 Point {
2     int x;
3     int y;
4     int z;
5 };
6
7 struct Body {
8     int tag;
9     struct Point points[3];
10 };
11
12 int dot(struct Point dp1, struct Point dp2) {
13     return dp1.x * dp2.x + dp1.y * dp2.y + dp1.z * dp2.z;
14 }
15
16 int sqrt(int si) {
17     int s_cnt = 0;
18     while (s_cnt <= si) {
19         int lower = s_cnt * s_cnt;
20         int upper = (s_cnt + 1) * (s_cnt + 1);
21         if ((si >= lower) && (si < upper)) {
22             return s_cnt;
23         }
24     }
```

```

24     s_cnt = s_cnt + 1;
25 }
26 return -1;
27 }
28
29 int dist(struct Point dip1, struct Point dip2) {
30     return sqrt(dot(dip1, dip2));
31 }
32
33 int main() {
34     struct Body bodies[3];
35     int o_cnt = 0;
36     int i_cnt = 0;
37     while (o_cnt < 3) {
38         i_cnt = 0;
39         while (i_cnt < 3) {
40             bodies[o_cnt].points[i_cnt].x = i_cnt * o_cnt + i_cnt;
41             bodies[o_cnt].points[i_cnt].y = i_cnt * o_cnt * o_cnt + i_cnt;
42             bodies[o_cnt].points[i_cnt].z = i_cnt * i_cnt * o_cnt * o_cnt +
                i_cnt;
43             i_cnt = i_cnt + 1;
44         }
45         bodies[o_cnt].tag = dist(bodies[o_cnt].points[1], bodies[o_cnt].
            points[2]);
46         o_cnt = o_cnt + 1;
47     }
48     write(bodies[0].tag + bodies[1].tag + bodies[2].tag);
49     return 0;
50 }

```

程序输入: 无; 预期输出: 19

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

```
1 int main() {
2     int len = 5;
3     int mat1[5][5];
4     int mat2[5][5];
5     int o_cnt = 0;
6     int i_cnt = 0;
7     int m_res = 0;
8     while (o_cnt < len) {
9         i_cnt = 0;
10        while (i_cnt < len) {
11            mat1[o_cnt][i_cnt] = i_cnt;
12            mat2[o_cnt][i_cnt] = o_cnt;
13            i_cnt = i_cnt + 1;
14        }
15        o_cnt = o_cnt + 1;
16    }
17
18    o_cnt = 0;
19    while (o_cnt < len) {
20        i_cnt = 0;
21        while (i_cnt < len) {
22            m_res = m_res + mat1[o_cnt][i_cnt] * mat2[o_cnt][i_cnt];
23            i_cnt = i_cnt + 1;
24        }
25        o_cnt = o_cnt + 1;
26    }
27    write(m_res);
28 }
```

程序输入: 无; 预期输出: 100

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 quick_sort(int arr[6], int left, int right)
2 {
3     int i, j, pivot;
4     i = left;
5     j = right;
6     pivot = arr[left];
7
8     if (i >= j) {
9         return 0;
10    }
11
12    while (i < j) {
13        while (i < j && arr[j] > pivot) {
14            j = j - 1;
15        }
16        if (i < j) {
17            arr[i] = arr[j];
18            i = i + 1;
19        }
20
21        while (i < j && arr[i] < pivot) {
22            i = i + 1;
23        }
24        if (i < j) {
25            arr[j] = arr[i];
26            j = j - 1;
27        }
28    }
```

```

29     arr[i] = pivot;
30     quick_sort(arr, left, i - 1);
31     quick_sort(arr, i + 1, right);
32     return 0;
33 }
34
35 int main() {
36     int len = 6;
37     int a[6];
38     int cnt = 0;
39     int sum = 0;
40     while (cnt < len) {
41         a[cnt] = read();
42         cnt = cnt + 1;
43     }
44
45     quick_sort(a, 0, len - 1);
46
47     cnt = 0;
48     while (cnt < len) {
49         write(a[cnt]);
50         cnt = cnt + 1;
51     }
52     return 0;
53 }

```

程序输入: 4 -1 100 24 50 -100 预期输出: -100 -1 4 24 50 100

程序输入: 1 -100 255 -300 500 -1000 预期输出: -1000 -300 -100 1 255 500

说明: 快速排序, 测试对于数组作为函数参数的翻译。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 get_max(int max_arr[4]) {
2      int max_idx = 0;
3      int max_val = max_arr[0];
4
5      int max_len = 4;
6      int max_cnt = 0;
7      while (max_cnt < max_len) {
8          if (max_arr[max_cnt] > max_val) {
9              max_idx = max_cnt;
10             max_val = max_arr[max_cnt];
11         }
12         max_cnt = max_cnt + 1;
13     }
14     return max_idx;
15 }
16
17 int mod(int m1, int m2) {
18     return m1 - (m1 / m2) * m2;
19 }
20
21 int main() {
22     int len = 4;
23     int val_arr[4][4];
24     int res_arr[4];
25     int copy_arr[2];
26     int o_cnt = 0;
27     int i_cnt = 0;
28
29     o_cnt = 0;
30     while (o_cnt < len) {
31         i_cnt = 0;
32         while (i_cnt < len) {

```

```
33     val_arr[o_cnt][i_cnt] = mod(o_cnt + i_cnt, len);
34     i_cnt = i_cnt + 1;
35 }
36 res_arr[o_cnt] = get_max(val_arr[o_cnt]);
37 o_cnt = o_cnt + 1;
38 }
39
40 copy_arr = res_arr;
41 write(copy_arr[0]);
42 write(copy_arr[1]);
43 return 0;
44 }
```

程序输入: 无; 预期输出: 3 2

说明: 测试对于较复杂的数组操作的翻译, 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问, 可以写邮件与[屈道涵](#)助教联系, 注意同时抄送给[许老师](#)。