

## 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 . . . . .	3
1.3	A-3 . . . . .	5
1.4	A-4 . . . . .	6
1.5	A-5 . . . . .	7
<b>2</b>	<b>B 组测试用例</b>	<b>7</b>
2.1	B-1 . . . . .	8
2.2	B-2 . . . . .	9
2.3	B-3 . . . . .	10
<b>3</b>	<b>C 组测试用例</b>	<b>11</b>
3.1	C-1 . . . . .	11
3.2	C-2 . . . . .	13
<b>4</b>	<b>E 组测试用例</b>	<b>15</b>
4.1	E1-1 . . . . .	15
4.2	E1-2 . . . . .	16
4.3	E1-3 . . . . .	17
4.4	E2-1 . . . . .	19
4.5	E2-2 . . . . .	20
4.6	E2-3 . . . . .	21
<b>5</b>	<b>结束语</b>	<b>23</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

#### 1.1.1 输入

```
1 int main() {
2     int a = 10, b = 20, c, d, e, f, g, h, i, j;
3
4     c = a + b;
5     d = b - a;
6     e = a * b;
7     f = b / a;
8
9     g = (b + a) * (b - a);
10    h = (a + b) * (c - d);
11    i = e / (f + g);
12    j = (a * b) + (c / d) - (e + f);
13
14    write(c);
15    write(d);
16    write(e);
17    write(f);
18    write(g);
19
20    write(h);
21    write(i);
22    write(j);
23
24    return 0;
25 }
```

#### 1.1.2 输出

```
1 >>> Empty
2 [30, 10, 200, 2, 300, 600, 0, 1]
```

---

### 1.1.3 说明

这个测试用例针对赋值与算术语句进行测试。

输出中以>>>起始的行表示程序输入，其后的一行表示程序输出。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（下同）。

## 1.2 A-2

### 1.2.1 输入

```
1  int main() {
2      int a, b, c, d, e;
3      int f = 60, g = 70;
4      a = read();
5      b = read();
6      c = read();
7      d = read();
8      e = read();
9
10     if (a < b) {
11         write(a);
12     }
13
14     if (c > b) {
15         write(b);
16     } else {
17         write(c);
18     }
19
20     if (d > a) {
21         if (d > b) {
22             write(d);
23         } else {
24             write(b);
25         }
26     } else {
27         write(a);
```

```
28     }
29
30     if (e < a)
31         write(1);
32     else if (e < b)
33         write(3);
34     else if (e < c)
35         write(5);
36     else if (e < d)
37         write(7);
38     else
39         write(9);
40
41     if ((f > a) && (g < b))
42         write(2);
43     else if ((f > a) || (g < b))
44         write(4);
45     else
46         write(6);
47
48     return 0;
49 }
```

### 1.2.2 输出

```
1 >>> -3, 46, 1, 75, 111
2 [-3, 1, 75, 9, 4]
3
4 >>> 35, 21, 2, -8, -31
5 [2, 35, 1, 4]
6
7 >>> 1, -7, 12, 85, 101
8 [-7, 85, 9, 4]
```

### 1.2.3 说明

主要针对分支语句进行测试。

## 1.3 A-3

### 1.3.1 输入

```
1 int main() {
2     int arr[10];
3     int start, end, result = 0;
4     int i;
5
6     i = 0;
7     while (i < 10) {
8         arr[i] = i * i;
9         i = i + 1;
10    }
11
12    start = read();
13    end = read();
14
15    i = start;
16    while (i <= end) {
17        result = result + arr[i];
18        i = i + 1;
19    }
20    write(result);
21
22    return 0;
23 }
```

### 1.3.2 输出

```
1 >>> 0, 1
2 [1]
3
4 >>> 5, 9
5 [255]
```

### 1.3.3 说明

主要测试一维数组。

## 1.4 A-4

### 1.4.1 输入

```
1 int calculateSum(int start, int end) {
2     int sum = 0;
3     int i = start;
4
5     while (i <= end) {
6         sum = sum + i;
7         i = i + 1;
8     }
9
10    return sum;
11 }
12
13 int main() {
14     int a, b, result;
15     a = read();
16     b = read();
17
18     result = calculateSum(a, b);
19     write(result);
20
21     return 0;
22 }
```

### 1.4.2 输出

```
1 >>> 1, -1
2 [0]
3
4 >>> 0, 0
5 [0]
6
7 >>> 1, 10
8 [55]
```

### 1.4.3 说明

主要测试循环语句。

## 1.5 A-5

### 1.5.1 输入

```
1 int multiplyNumbers(int n) {
2     if (n >= 1)
3         return n * multiplyNumbers(n - 1);
4     else
5         return 1;
6 }
7
8 int main() {
9     int x;
10    x = read();
11    write(multiplyNumbers(x));
12    return 0;
13 }
```

### 1.5.2 输出

```
1 >>> 0
2 [1]
3
4 >>> 6
5 [720]
```

### 1.5.3 说明

主要测试函数调用。

## 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

### 2.1.1 输入

```
1 int binaryToDecimal(int binary) {
2     int decimal = 0;
3     int multiplier = 1;
4     while (binary != 0) {
5         decimal = decimal + (binary - (binary / 10) * 10) * multiplier;
6         binary = binary / 10;
7         multiplier = multiplier * 2;
8     }
9     return decimal;
10 }
11
12 int calculate_power(int base, int power) {
13     int i = 0, result = 1;
14     while (i < power) {
15         result = result * base;
16         i = i + 1;
17     }
18     return result;
19 }
20
21 int decimal_to_binary(int d) {
22     int digitCount = 1;
23     int j = digitCount - 1;
24     int temp = d;
25     while (temp >= 2) {
26         temp = temp / 2;
27         digitCount = digitCount + 1;
28     }
29     while (j >= 0) {
30         if (d >= calculate_power(2, j)) {
31             write(1);
32             d = d - calculate_power(2, j);
33         } else
34             write(0);
```



```

35     j = j - 1;
36 }
37 return 0;
38 }
39
40 int main() {
41     int bin;
42     int dec;
43
44     bin = read();
45     dec = binaryToDecimal(bin);
46     write(dec);
47
48     dec = read();
49     decimal_to_binary(dec);
50
51     return 0;
52 }

```

### 2.1.2 输出

```

1 >>> 1101, 16
2 [13, 1]

```

## 2.2 B-2

### 2.2.1 输入

```

1 int sumOfDigits(int num) {
2     int digit;
3     int sum = 0;
4     while (num > 0) {
5         digit = num - (num / 10) * 10;
6         sum = sum + digit;
7         num = num / 10;
8     }
9     return sum;
10 }

```

```

11
12 int sumOfSumsOfDigits(int s, int e) {
13     int totalSum = 0, i = s;
14     while (i <= e) {
15         totalSum = totalSum + sumOfDigits(i);
16         i = i + 1;
17     }
18     return totalSum;
19 }
20
21 int main() {
22     int start, end, result;
23     start = read();
24     end = read();
25     result = sumOfSumsOfDigits(start, end);
26     write(result);
27     return 0;
28 }

```

## 2.2.2 输出

```

1 >>> 10, 20
2 [57]

```

## 2.3 B-3

### 2.3.1 输入

```

1
2 int main() {
3     int a[10], b[10], c[10], n;
4     int i;
5     n = read();
6
7     i = 0;
8     while (i < n) {
9         a[i] = read();
10        i = i + 1;

```

```

11     }
12     i = 0;
13     while (i < n) {
14         b[i] = read();
15         i = i + 1;
16     }
17
18     i = 0;
19     while (i < n) {
20         c[i] = a[i] * b[i];
21         i = i + 1;
22     }
23
24     i = 0;
25     while (i < n) {
26         write(c[i]);
27         i = i + 1;
28     }
29
30     return 0;
31 }

```

### 2.3.2 输出

```

1 >>> 2, 1, 1, 1, 1
2 [1, 1]
3
4 >>> 4, 3, 2, 1, 0, 2, 4, 5, 7
5 [6, 8, 5, 0]

```

## 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

#### 3.1.1 输入

```

1  int min(int a, int b) {
2      if (a <= b)
3          return a;
4      else
5          return b;
6  }
7
8  int jumpSearch() {
9      int arr[16];
10     int x;
11     int n = 16;
12     int step = 4;
13     int i = 0;
14     int prev = 0;
15
16     while (i < 16) {
17         arr[i] = read();
18         i = i + 1;
19     }
20     x = read();
21
22     while (arr[min(step, n) - 1] < x) {
23         prev = step;
24         step = step + 4;
25         if (prev >= n)
26             return -1;
27     }
28
29     while (arr[prev] < x) {
30         prev = prev + 1;
31
32         if (prev == min(step, n))
33             return -1;
34     }
35     if (arr[prev] == x)
36         return prev;
37

```

```

38     return -1;
39 }
40
41 int main() {
42     int index = jumpSearch();
43     write(index);
44     return 0;
45 }

```

### 3.1.2 输出

```

1 >>> 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 55
2 [10]
3
4 >>> 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 610
5 [15]

```

## 3.2 C-2

### 3.2.1 输入

```

1 int min(int a, int b) {
2     if (a <= b)
3         return a;
4     else
5         return b;
6 }
7
8 int fibMonaccianSearch() {
9     int arr[16];
10    int x;
11    int i = 0;
12    int n = 16;
13    int offset = -1;
14
15    int fibMMm2 = 0;
16    int fibMMm1 = 1;
17    int fibM = fibMMm2 + fibMMm1;

```

```

18
19 while (i < 16) {
20     arr[i] = read();
21     i = i + 1;
22 }
23 x = read();
24
25 while (fibM < n) {
26     fibMMm2 = fibMMm1;
27     fibMMm1 = fibM;
28     fibM = fibMMm2 + fibMMm1;
29 }
30
31 while (fibM > 1) {
32     i = min(offset + fibMMm2, n - 1);
33
34     if (arr[i] < x) {
35         fibM = fibMMm1;
36         fibMMm1 = fibMMm2;
37         fibMMm2 = fibM - fibMMm1;
38         offset = i;
39     }
40
41     else if (arr[i] > x) {
42         fibM = fibMMm2;
43         fibMMm1 = fibMMm1 - fibMMm2;
44         fibMMm2 = fibM - fibMMm1;
45     }
46
47     else
48         return i;
49 }
50
51 if (fibMMm1 && arr[offset + 1] == x)
52     return offset + 1;
53
54 return -1;

```

```

55 }
56
57 int main() {
58     write(fibMonaccianSearch());
59     return 0;
60 }

```

### 3.2.2 输出

```

1 >>> 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 55
2 [10]
3
4 >>> 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 610
5 [15]

```

## 4 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。

### 4.1 E1-1

#### 4.1.1 输入

```

1
2 struct Activity {
3     int start;
4     int finish;
5 };
6
7 int main() {
8     struct Activity a1, a2;
9     a1.start = read();
10    a1.finish = read();
11    a2.finish = a1.start;
12    a2.start = a1.finish;
13    write(a1.start);

```

```
14     write(a1.finish);
15     write(a2.start);
16     write(a2.finish);
17     return 0;
18 }
```

### 4.1.2 输出

```
1 >>> 1, 2
2 [1, 2, 2, 1]
```

### 4.1.3 说明

测试对于简单结构体的翻译。针对选做**要求 4.1**的同学，其他同学需要提示无法翻译且不输出中间代码。

## 4.2 E1-2

### 4.2.1 输入

```
1 struct Rectangle {
2     int length;
3     int width;
4 };
5
6 int calculateArea(struct Rectangle rect) { return rect.length * rect.
    width; }
7
8 int main() {
9     int numRectangles;
10    int totalArea = 0;
11    struct Rectangle r;
12
13    numRectangles = read();
14    while (numRectangles > 0) {
15        r.length = read();
16        r.width = read();
17        totalArea = totalArea + calculateArea(r);
```



```

18     numRectangles = numRectangles - 1;
19 }
20
21 write(totalArea);
22 return 0;
23 }

```

#### 4.2.2 输出

```

1 >>> 2, 3, 4, 7, 8
2 [68]

```

#### 4.2.3 说明

测试将结构体作为函数参数。针对选做**要求 4.1**的同学，其他同学需要提示无法翻译且不输出中间代码。

### 4.3 E1-3

#### 4.3.1 输入

```

1
2 struct Point {
3     int x;
4     int y;
5 };
6
7 struct Circle {
8     struct Point center;
9     int radius;
10 };
11
12 int calculateCircleArea(struct Circle circle) {
13     return 3 * circle.radius * circle.radius;
14 }
15
16 int sumOfAreasPlusDistanceSquared(struct Circle circle1,
17                                   struct Circle circle2) {

```

```

18  int area1 = calculateCircleArea(circle1);
19  int area2 = calculateCircleArea(circle2);
20  int sumOfAreas = area1 + area2;
21
22  int dx = circle1.center.x - circle2.center.x;
23  int dy = circle1.center.y - circle2.center.y;
24  int distance = dx * dx + dy * dy;
25
26  return sumOfAreas + distance;
27 }
28
29 int main() {
30     struct Circle c1, c2;
31
32     c1.center.x = read();
33     c1.center.y = read();
34     c1.radius = read();
35
36     c2.center.x = read();
37     c2.center.y = read();
38     c2.radius = read();
39
40     write(sumOfAreasPlusDistanceSquared(c1, c2));
41
42     return 0;
43 }

```

### 4.3.2 输出

```

1 >>> 1, 3, 5, 2, 4, 6
2 [185]

```

### 4.3.3 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。针对选做**要求 4.1**的同学，其他同学需要提示无法翻译且不输出中间代码。

## 4.4 E2-1

### 4.4.1 输入

```
1  int main() {
2      int n = 3;
3      int magicSquare[3][3];
4      int i = n / 2;
5      int j = n - 1;
6      int num = 1;
7
8      while (num <= n * n) {
9          if (i == -1 && j == n) {
10             j = n - 2;
11             i = 0;
12         } else {
13             if (j == n)
14                 j = 0;
15             if (i < 0)
16                 i = n - 1;
17         }
18         if (magicSquare[i][j]) {
19             j = j - 2;
20             i = i + 1;
21         } else {
22             magicSquare[i][j] = num;
23             num = num + 1;
24             j = j + 1;
25             i = i - 1;
26         }
27     }
28     write(n);
29     write(n * (n * n + 1) / 2);
30     i = 0;
31     j = 0;
32     while (i < n) {
33         while (j < n) {
34             write(magicSquare[i][j]);
```

```

35     j = j + 1;
36 }
37 i = i + 1;
38 }
39 return 0;
40 }

```

#### 4.4.2 输出

```

1 >>> Empty
2 [3, 15, 2, 7, 6]

```

#### 4.4.3 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。针对选做**要求 4.2**的同学，其他同学需要提示无法翻译且不输出中间代码。

### 4.5 E2-2

#### 4.5.1 输入

```

1 int maxSumarray(int a[10], int s) {
2     int j = 0;
3     int max_sum_so_far = 0;
4     int max_ending_here = 0;
5
6     while (j < s) {
7         max_ending_here = max_ending_here + a[j];
8
9         if (max_ending_here < 0) {
10             max_ending_here = 0;
11         }
12         if (max_sum_so_far < max_ending_here) {
13             max_sum_so_far = max_ending_here;
14         }
15         j = j + 1;
16     }
17     return max_sum_so_far;

```

```

18 }
19
20 int main() {
21
22     int i = 0, size = 10;
23     int arr[10];
24     while (i < size) {
25         arr[i] = read();
26         i = i + 1;
27     }
28     write(maxSumarray(arr, size));
29     return 0;
30 }

```

#### 4.5.2 输出

```

1 >>> 1, 1, 2, 3, 5, 8, 13, 21, 34, 55
2 [143]

```

#### 4.5.3 说明

测试对于数组作为函数参数的翻译。针对选做**要求 4.2**的同学，其他同学需要提示无法翻译且不输出中间代码。

### 4.6 E2-3

#### 4.6.1 输入

```

1 int minDistance(int n, int visited[10], int d[10]) {
2     int min = 2147483647, min_vertex;
3     int v = 0;
4     while (v < n) {
5         if (!visited[v] && d[v] <= min) {
6             min = d[v];
7             min_vertex = v;
8         }
9         v = v + 1;
10    }

```

```

11     return min_vertex;
12 }
13
14 int main() {
15     int i, j, k;
16     int graph[10][10], isvisited[10], dist[10];
17     int sizeOfGraph, Edges, minV;
18     sizeOfGraph = read();
19     i = 0;
20     j = 0;
21     while (i < sizeOfGraph) {
22         while (j < sizeOfGraph) {
23             graph[i][j] = 0;
24             j = j + 1;
25         }
26         i = i + 1;
27     }
28     Edges = read();
29     i = 0;
30     while (i < Edges) {
31         j = read();
32         k = read();
33         graph[j][k] = read();
34         graph[k][j] = graph[j][k];
35         i = i + 1;
36     }
37
38     i = 0;
39     while (i < sizeOfGraph) {
40         dist[i] = 2147483647;
41         isvisited[i] = 0;
42         i = i + 1;
43     }
44
45     dist[0] = 0;
46
47     i = 0;

```

```

48 while (i < sizeofGraph - 1) {
49     minV = minDistance(sizeofGraph, isvisited, dist);
50     isvisited[minV] = 1;
51
52     j = 0;
53     while (j < sizeofGraph) {
54         if (isvisited[j] == 0 && graph[minV][j] &&
55             dist[minV] + graph[minV][j] < dist[j])
56             dist[j] = dist[minV] + graph[minV][j];
57         j = j + 1;
58     }
59     i = i + 1;
60 }
61
62 i = 0;
63 while (i < sizeofGraph) {
64     write(dist[i]);
65     i = i + 1;
66 }
67 return 0;
68 }

```

#### 4.6.2 输出

```

1 >>> 4, 5, 0, 1, 2, 0, 2, 1, 1, 2, 6, 1, 3, 1, 2, 3, 3
2 [0, 2, 1, 3]

```

#### 4.6.3 说明

测试对于较复杂的数组及其作为函数参数进行函数的调用。针对选做**要求 4.2**的同学，其他同学需要提示无法翻译且不输出中间代码。

## 5 结束语

若对本文档有任何疑议，可写邮件与周意可助教联系，注意同时抄送给许畅老师。