

## 编译原理第二次实验测试用例：目录

<b>1</b>	<b>A 组测试用例</b>	<b>3</b>
1.1	A-1 . . . . .	3
1.2	A-2 . . . . .	4
1.3	A-3 . . . . .	4
1.4	A-4 . . . . .	5
1.5	A-5 . . . . .	6
1.6	A-6 . . . . .	7
1.7	A-7 . . . . .	7
1.8	A-8 . . . . .	8
1.9	A-9 . . . . .	9
1.10	A-10 . . . . .	9
1.11	A-11 . . . . .	10
1.12	A-12 . . . . .	11
1.13	A-13 . . . . .	11
1.14	A-14 . . . . .	12
1.15	A-15 . . . . .	14
1.16	A-16 . . . . .	14
1.17	A-17 . . . . .	15
1.18	A-18 . . . . .	16
1.19	A-19 . . . . .	16
1.20	A-20 . . . . .	17
<b>2</b>	<b>B 组测试用例</b>	<b>18</b>
2.1	B-1 . . . . .	18
2.2	B-2 . . . . .	20
<b>3</b>	<b>C 组测试用例</b>	<b>21</b>
3.1	C-1 . . . . .	21
3.2	C-2 . . . . .	22
<b>4</b>	<b>D 组测试用例</b>	<b>23</b>
4.1	D-1 . . . . .	23
4.2	D-2 . . . . .	25
4.3	D-3 . . . . .	26

<b>5</b>	<b>E 组测试用例</b>	<b>28</b>
5.1	E-1 . . . . .	28
5.2	E-2 . . . . .	29
5.3	E-3 . . . . .	30
<b>6</b>	<b>结束语</b>	<b>31</b>

评分时每一行至多只检查一个需要计分的本质错误；由本质错误引发的次生错误可报可不报。错误编号和行号之后的说明文字仅供理解，不作为评分依据。

## 1 A 组测试用例

本组测试用例共 20 个，测试用例 1-17 分别对应语义错误 1-17，之后三个测试用例对应于语义错误 7, 9, 15。每个用例仅在其中一行含有语义错误。某些语义错误可能会产生连锁反应。测试用例 A-i 对应的“本质错误”的错误类型是必须报出来的，如果报出其他错误，只要是由本质错误连带引发的（包括但不限于下面明确给出的情况），我们都不会扣分。错误编号和行号之后的说明文字不要求与给出的输出完全一致，仅供助教理解使用，不作为评分依据。

### 1.1 A-1

#### 1.1.1 输入

```
1 struct Record {
2     int key;
3     float value;
4 };
5
6 int process() {
7     struct Record r1, r2;
8     r1.key = 42;
9     r1.value = 2.71;
10    r2.key = 100;
11    r2.key = r2.key + 1;
12    r2.key = r2.key + 1;
13    r2.key = r2.key + 1;
14    r2.key = r2.key + 1;
15    r2.value = margin_rate;
16    return 0;
17 }
```

#### 1.1.2 输出

```
1 Error type 1 at Line 15: Undefined Variable.
```

### 1.1.3 说明

错误类型 1: 第 15 行使用未定义变量 `margin_rate`。可以多报一个 5 型错误。

## 1.2 A-2

### 1.2.1 输入

```
1 struct Node {
2     int val;
3     float weight;
4 };
5
6 int main() {
7     struct Node n;
8     n.val = 7;
9     n.weight = 0.88;
10    n.val = n.val + 1;
11    n.val = n.val + 1;
12    n.val = n.val + 1;
13    n.val = n.val + 1;
14    n.val = fetch_value(n.val);
15    return 0;
16 }
```

### 1.2.2 输出

```
1 Error type 2 at Line 14: Undefined function.
```

### 1.2.3 说明

错误类型 2: 第 14 行调用未定义函数。可以多报一个 5 型错误

## 1.3 A-3

### 1.3.1 输入

```
1 struct Entry {
2     int code;
3     float ratio;
```

```
4 };
5
6
7 int run() {
8     int idx, sum;
9     int idx;
10    return sum;
11 }
```

### 1.3.2 输出

```
1 Error type 3 at Line 9: Redefined Variable.
```

### 1.3.3 说明

错误类型 3：第 9 行同一作用域内变量重复定义。

## 1.4 A-4

### 1.4.1 输入

```
1 struct Pair {
2     int left;
3     int right;
4 };
5
6 int merge(int a, int b) {
7     return a + b;
8 }
9
10 int main() {
11     int x = 3, y = 5;
12     int z = merge(x, y);
13     return z;
14 }
15
16 int merge(int p) {
17     return p;
```

```
18 }
```

### 1.4.2 输出

```
1 Error type 4 at Line 16: Redefined Function.
```

### 1.4.3 说明

错误类型 4：第 16 行函数重复定义。也可以报在第 6 行。

## 1.5 A-5

### 1.5.1 输入

```
1 struct Metric {  
2     int count;  
3     float rate;  
4 };  
5  
6 int main() {  
7     struct Metric m;  
8     int threshold;  
9     m.count = 10;  
10    m.rate = 0.95;  
11    threshold = m.rate;  
12    return threshold;  
13 }
```

### 1.5.2 输出

```
1 Error type 5 at Line 11: Type mismatched for assignment.
```

### 1.5.3 说明

错误类型 5：第 11 行赋值号两侧类型不匹配（如 float 赋给 int）。

## 1.6 A-6

### 1.6.1 输入

```
1 struct Box {
2     float width;
3     float height;
4 };
5
6 int main() {
7     struct Box b;
8     float area = 2.5;
9     b.width = 3.0;
10    b.height = 4.0;
11    b.width * b.height = area;
12    return 0;
13 }
```

### 1.6.2 输出

```
1 Error type 6 at Line 11: The left-hand side of an assignment must
   be a variable.
```

### 1.6.3 说明

错误类型 6：第 11 行赋值号左侧不是合法左值。

## 1.7 A-7

### 1.7.1 输入

```
1 struct Cell {
2     int row;
3     int col;
4 };
5
6 int main() {
7     struct Cell c;
8     float ratio = 1.5;
```

```

9      c.row = 2;
10     c.col = 3;
11     c.row = c.row + ratio;
12     return 0;
13 }

```

### 1.7.2 输出

```

1 Error type 7 at Line 11: Type mismatched for operands.

```

### 1.7.3 说明

错误类型 7：第 11 行操作数类型与运算符不匹配。可以多报一个 5 型错误。

## 1.8 A-8

### 1.8.1 输入

```

1 struct Result {
2     int code;
3 };
4
5 float get_ratio() {
6     int x = 10;
7     return x;
8 }
9
10 int main() {
11     float r = get_ratio();
12     return 0;
13 }

```

### 1.8.2 输出

```

1 Error type 8 at Line 7: Type mismatched for return.

```

### 1.8.3 说明

错误类型 8：第 7 行 `return` 表达式类型与函数声明的返回类型不一致。



## 1.9 A-9

### 1.9.1 输入

```
1 struct Tuple {
2     int first;
3     int second;
4 };
5
6 int add_pair(int a, int b) {
7     return a + b;
8 }
9
10 int main() {
11     struct Tuple t;
12     int s;
13     t.first = 1;
14     t.second = 2;
15     s = add_pair(t.first, t.second, 99);
16     return s;
17 }
```

### 1.9.2 输出

```
1 Error type 9 at Line 15: Function is not applicable for arguments.
```

### 1.9.3 说明

错误类型 9：第 15 行函数调用时实参与形参数目或类型不匹配。

## 1.10 A-10

### 1.10.1 输入

```
1 int partition(int arr[10], int low, int high) {
2     int pivot = arr[low];
3     int i = low;
4     int j = high;
5     while (i < j) {
```

```

6         while (arr[i] <= pivot) {
7             i = i + 1;
8         }
9         pivot[0] = arr[j];
10        j = j - 1;
11    }
12    return i;
13 }
14
15 int main() {
16     int a[5];
17     return 0;
18 }

```

### 1.10.2 输出

```

1 Error type 10 at Line 9: Not an array.

```

### 1.10.3 说明

错误类型 10: 第 9 行对非数组类型变量使用 [] 下标。

## 1.11 A-11

### 1.11.1 输入

```

1 struct Handler {
2     int id;
3     float factor;
4 };
5
6 int main() {
7     struct Handler h;
8     h.id = 1;
9     h.factor = 2.5;
10    h(10);
11    return 0;
12 }

```

### 1.11.2 输出

```
1 Error type 11 at Line 10: Not a function.
```

### 1.11.3 说明

错误类型 11: 第 10 行对非函数变量使用 () 函数调用操作符。

## 1.12 A-12

### 1.12.1 输入

```
1 int main() {  
2     int buf[20];  
3     float idx = 2.7;  
4     buf[idx] = 100;  
5     return 0;  
6 }
```

### 1.12.2 输出

```
1 Error type 12 at Line 4: Not an integer.
```

### 1.12.3 说明

错误类型 12: 第 4 行数组 [] 下标为非整数类型。

## 1.13 A-13

### 1.13.1 输入

```
1 struct Vertex {  
2     float coord_x;  
3     float coord_y;  
4 };  
5  
6 float normSquared(struct Vertex v1, struct Vertex v2) {  
7     float dx = v1.coord_x - v2.coord_x;  
8     float dy = v1.coord_y - v2.coord_y;  
9     return dx * dx + dy * dy;  
}
```

```

10 }
11
12 int main() {
13     struct Vertex pt_a;
14     struct Vertex pt_b;
15     int iter = 0;
16     float dist;
17
18     pt_a.coord_x = 1.0;
19     pt_a.coord_y = 2.0;
20     pt_b.coord_x = 4.0;
21     pt_b.coord_y = 6.0;
22
23     while (iter < 3) {
24         if (iter / 2 == 0) {
25             dist = normSquared(pt_a, pt_b);
26         } else {
27             float tmp = dist;
28             tmp.member = tmp * 0.5;
29         }
30         iter = iter + 1;
31     }
32
33     return 0;
34 }

```

### 1.13.2 输出

```

1 Error type 13 at Line 28: Illegal use of ".".

```

### 1.13.3 说明

错误类型 13: 第 28 行对非结构体类型变量使用. 成员访问。

## 1.14 A-14

### 1.14.1 输入

```

1 struct Node {
2     float weight;
3     int flag;
4 };
5
6 int evalNode(struct Node n) {
7     if (n.flag > 0) {
8         return 1;
9     } else {
10        return n.err_field;
11    }
12 }
13
14 int main() {
15     struct Node nd;
16     int res;
17     nd.weight = 36.5;
18     nd.flag = 1;
19
20     res = evalNode(nd);
21
22     while (res == 0) {
23         nd.flag = nd.flag + 1;
24         res = evalNode(nd);
25     }
26
27     return 0;
28 }

```

### 1.14.2 输出

```

1 Error type 14 at Line 10: Not-existen field.

```

### 1.14.3 说明

错误类型 14：第 10 行访问结构体中未定义的域。

## 1.15 A-15

### 1.15.1 输入

```
1 struct Cell {  
2     int tag;  
3     float val;  
4     int tag;  
5     int slot[16];  
6 };  
7  
8 int main() {  
9     struct Cell c;  
10    int x;  
11    c.tag = 1;  
12    c.val = 2.5;  
13  
14    return 0;  
15 }
```

### 1.15.2 输出

```
1 Error type 15 at Line 4: Redefined field.
```

### 1.15.3 说明

错误类型 15: 结构体中域名重复定义, 报错位于第 4 行或第 2 行 (二者择一)。

## 1.16 A-16

### 1.16.1 输入

```
1 struct Box {  
2     float w;  
3     float h;  
4 };  
5  
6 struct Box {  
7     int id;
```

```

8      int len;
9  };
10
11 int main() {
12     struct Box b;
13     b.w = 1.5;
14     b.h = 2.5;
15
16     return 0;
17 }

```

### 1.16.2 输出

```

1 Error type 16 at Line 6: Duplicated name.

```

### 1.16.3 说明

错误类型 16: 第 6 行结构体类型名与已定义过的结构体重复。

## 1.17 A-17

### 1.17.1 输入

```

1 struct KnownType {
2     int val;
3 };
4
5 int main() {
6     struct UnknownType u;
7     struct KnownType k;
8     k.val = 10;
9     return 0;
10 }

```

### 1.17.2 输出

```

1 Error type 17 at Line 6: Undefined structure3.

```

### 1.17.3 说明

错误类型 17: 第 6 行使用未定义的结构体类型定义变量。

## 1.18 A-18

### 1.18.1 输入

```
1 struct Item {
2     int code;
3     struct {
4         float mass = 2.0;
5         int cells[3];
6     } spec;
7 };
8
9 int main() {
10     struct Item it;
11     it.code = 100;
12     return 0;
13 }
```

### 1.18.2 输出

```
1 Error type 15 at Line 4: Field can't be initied.
```

### 1.18.3 说明

错误类型 15: 第 4 行在结构体定义中对域进行初始化。

## 1.19 A-19

### 1.19.1 输入

```
1 struct Grid {
2     int mat[3][3];
3     int nr;
4     int nc;
5 };
6
```



```

7  int main() {
8      struct Grid g1;
9      int vec[3];
10
11     g1.nr = 3;
12     g1.nc = 3;
13
14     g1.mat[0][0] = vec + g1.nr;
15
16     return 0;
17 }

```

### 1.19.2 输出

```

1 Error type 7 at Line 14: Type mismatched for operands.

```

### 1.19.3 说明

错误类型 7：第 14 行操作数类型不匹配（数组与整数参与运算）。

## 1.20 A-20

### 1.20.1 输入

```

1  struct Node {
2      float data;
3  };
4
5  float read_node(struct Node n) {
6      return n.data;
7  }
8
9  int main() {
10     struct Node nd;
11     nd.data = 1.5;
12     return read_node(nd);
13 }

```

### 1.20.2 输出

```
1 Error type 8 at Line 12: Type mismatched for return.
```

### 1.20.3 说明

错误类型 8：第 12 行 `return` 表达式类型与函数返回类型不一致。

## 2 B 组测试用例

本组测试用例共 2 个，其中包含多个语义错误。每一行至多包含一个需要计分的本质错误；由该错误导致的次生错误可报可不报。同一个语义错误可能会有连锁反应，其处理方式与 A 类用例相同，只要是合理的（包括但不限于下面明确给出的情况），都不会影响得分。

### 2.1 B-1

#### 2.1.1 输入

```
1 struct Vec2 {
2     int x;
3     int y;
4 };
5
6 int distSq(struct Vec2 v1, struct Vec2 v2) {
7     int dx = v1.x - v2.x;
8     int dy = v1.y - v2.y;
9     return dx * dx + dy * dy;
10 }
11
12 int main() {
13     struct Vec2 p1, p2;
14     int arr[5];
15     int i = 0;
16     int sum = 0;
17
18     p1.x = 3;
19     p1.y = 4;
20     p2.x = 6;
21     p2.y = 8;
```

```

22
23     while (i < 5) {
24         sum = sum + arr[i];
25         i = i + 1;
26     }
27
28     sum = arr + p1;
29
30     distSq(p1, p2) = sum;
31
32     if (sum > 10) {
33         p1.x = p1.x + 1;
34     } else {
35         p1.x = p1.x - 1;
36     }
37
38     sum = sum.x;
39
40     p1.y = p1.y + 2;
41
42     sum = p1.z;
43
44     return sum;
45 }

```

### 2.1.2 输出

```

1 Error type 7 at Line 28: Type mismatched for operands.
2 Error type 6 at Line 30: The left-hand side of an assignment must
  be a variable.
3 Error type 13 at Line 38: Illegal use of ".".
4 Error type 14 at Line 42: Not-existen field.

```

### 2.1.3 说明

B 类综合用例：第 28、30、38、42 行分别含类型 7、6、13、14 的语义错误。

## 2.2 B-2

### 2.2.1 输入

```
1 struct ScoreRecord {
2     int id;
3     int score;
4 };
5
6 int sumScores(struct ScoreRecord a, struct ScoreRecord b, struct
    ScoreRecord c) {
7     return a.score + b.score + c.score;
8 }
9
10 int topId(struct ScoreRecord x, struct ScoreRecord y) {
11     int bid = x.id;
12     int hs = x.score;
13
14     if (y.score > hs) {
15         hs = y.score;
16         bid = y.id;
17     }
18
19     return bid;
20 }
21
22 int showResult(int score_total, int best_id) {
23     if (score_total + 0.5 > 90) {
24         best_id[0] = 5;
25     }
26
27     return 0.0;
28 }
29
30 int main() {
31     struct ScoreRecord r1, r2, r3;
32     int total;
33     int tid;
```

```

34
35     r1.id = 101;
36     r1.score = 85;
37     r2.id = 102;
38     r2.score = 92;
39     r3.id = 103;
40     r3.score = 78;
41
42     total = sumScores(r1, r2, r3);
43
44     tid = topId(r1, r2);
45
46     showResult(total, tid);
47
48     return total;
49 }

```

### 2.2.2 输出

```

1 Error type 7 at Line 23: Type mismatched for operands.
2 Error type 10 at Line 24: Not an array.
3 Error type 8 at Line 27: Type mismatched for return.

```

### 2.2.3 说明

B 类综合用例：判题须包含第 23、24、27 行的错误类型 7、10、8。

## 3 C 组测试用例

本组测试用例共 2 个，不包含任何错误。

### 3.1 C-1

#### 3.1.1 输入

```

1 int add(int a, int b) {
2     return a + b;
3 }
4

```

```
5 int main() {
6     int x;
7     int y;
8     x = 3;
9     y = 5;
10    return add(x, y);
11 }
```

### 3.1.2 输出

---

### 3.1.3 说明

无语义错误的合法 C- 程序（简单加法与函数调用）。

## 3.2 C-2

### 3.2.1 输入

```
1 int fib(int n) {
2     if (n <= 1) {
3         return n;
4     }
5     return fib(n - 1) + fib(n - 2);
6 }
7
8 int main() {
9     int i = 0;
10    int result;
11    while (i < 10) {
12        result = fib(i);
13        i = i + 1;
14    }
15    return result;
16 }
```

### 3.2.2 输出

---

### 3.2.3 说明

无语义错误的合法 C- 程序 (Fibonacci 递归)。

## 4 D 组测试用例

本组测试用例共 3 个，针对不同分组进行测试。需要能够识别其语言特性，如果提示错误则不得分；其他分组的同学需要识别出其中的错误，如果没有报错，则将视为违规，将会倒扣分。

### 4.1 D-1

#### 4.1.1 输入

```
1 struct Result {
2     int sum_val;
3     int max_val;
4 };
5
6 int sumArr(int sum_decl_arr[3], int sum_decl_count);
7
8 int sumArr(int sum_def_arr[3], int sum_def_count) {
9     int sum_idx = 0, sum_acc = 0;
10    while (sum_idx < sum_def_count) {
11        sum_acc = sum_acc + sum_def_arr[sum_idx];
12        sum_idx = sum_idx + 1;
13    }
14    return sum_acc;
15 }
16
17 int maxArr(int max_def_arr[3], int max_def_count) {
18     int max_idx = 0, max_best = max_def_arr[0];
19     while (max_idx < max_def_count) {
20         if (max_def_arr[max_idx] > max_best) {
21             max_best = max_def_arr[max_idx];
22         }
23         max_idx = max_idx + 1;
24     }
25     return max_best;
26 }
```

```

27
28 struct Result compute(int comp_left, int comp_right, int comp_arr
    [3], int comp_count) {
29     struct Result comp_result;
30     comp_result.sum_val = sumArr(comp_arr, comp_count);
31     comp_result.max_val = maxArr(comp_arr, comp_count);
32     if (comp_left > comp_right) {
33         comp_result.max_val = comp_result.max_val + comp_left;
34     } else {
35         comp_result.sum_val = comp_result.sum_val + comp_right;
36     }
37     return comp_result;
38 }
39
40 int main() {
41     struct Result main_result;
42     int main_arr[3];
43     int main_x, main_y;
44     int main_idx = 0;
45
46     while (main_idx < 3) {
47         main_arr[main_idx] = main_idx * 5 + 2;
48         main_idx = main_idx + 1;
49     }
50
51     main_x = main_arr[0] + 1;
52     main_y = main_arr[1] + 3;
53
54     main_result = compute(main_x, main_y, main_arr, 3);
55
56     if (main_result.sum_val > main_result.max_val) {
57         return main_result.sum_val;
58     } else {
59         return main_result.max_val;
60     }
61 }
62

```



```
63 int maxArr(int max_decl_arr[3], int max_decl_count);
```

#### 4.1.2 输出

```
1 Error type B at Line 6: Syntax error.  
2 Error type B at Line 63: Syntax error.
```

#### 4.1.3 说明

含函数前置声明的程序。分组 0、1 完成 3.1 时无输出；分组 2、3 未实现函数声明时在第 6、63 行报语法错误 B。各函数声明、定义和局部变量使用不同名称，避免额外变量重定义错误干扰。

### 4.2 D-2

#### 4.2.1 输入

```
1 struct Pt {  
2     int x;  
3     int y;  
4 };  
5  
6 float area(int r) {  
7     float pi = 3.14159;  
8     return pi;  
9 }  
10  
11 int main() {  
12     int cnt = 0;  
13     int tot = 0;  
14  
15     struct Pt p;  
16     int buf[3];  
17  
18     {  
19         int cnt = 5;  
20         float tot = area(cnt);  
21         p.x = cnt;
```

```

22
23     while (cnt > 0) {
24         int buf = 99;
25         cnt = cnt - 1;
26     }
27 }
28
29 buf[0] = 99;
30 return 0;
31 }

```

## 4.2.2 输出

```

1 Error type 3 at Line 19: Redefined variable or structure conflict.
2 Error type 3 at Line 20: Redefined variable or structure conflict.
3 Error type 3 at Line 24: Redefined variable or structure conflict.

```

## 4.2.3 说明

嵌套作用域与变量重定义。分组 0、2 完成 3.2 时无输出；分组 1、3 在第 19、20、24 行报类型 3（变量/作用域冲突）。

## 4.3 D-3

### 4.3.1 输入

```

1 struct T1 { int t1_a; float t1_b; } v1;
2 struct T2 { int t2_c; float t2_d; } v2;
3
4 struct M1 { int m1_arr[3][3]; } m1;
5 struct M2 { int m2_arr[2][4]; } m2;
6
7 struct Payload {
8     int payload_vals[3];
9     struct { int payload_meta_id; float payload_meta_sc; }
10     payload_meta;
11 };

```

```

12 struct Record {
13     int record_nums[4];
14     struct {
15         int record_det_id;
16         float record_det_sc;
17     } record_det;
18 };
19
20 struct Record fetch(struct Payload payload_arg) {
21     return payload_arg;
22 }
23
24 int main() {
25     float tot;
26     struct Payload payload_obj;
27     struct Record record_obj;
28
29     v1 = v2;
30
31     m1 = m2;
32
33     record_obj = fetch(payload_obj);
34
35     return 0;
36 }

```

### 4.3.2 输出

```

1 Error type 8 at Line 21: Type mismatched for return.
2 Error type 5 at Line 29: Type mismatched for assignment.
3 Error type 5 at Line 31: Type mismatched for assignment.

```

### 4.3.3 说明

结构体名等价与结构等价。分组 0、3 完成 3.3 时无输出；分组 1、2 在第 21、29、31 行报类型 8 与 5（返回类型及赋值不匹配）。字段名、形参名和局部变量名均避免与其他符号重复。

## 5 E 组测试用例

本组测试用例共 3 个，针对不同分组进行测试。

### 5.1 E-1

#### 5.1.1 输入

```
1 struct A { int x; int y; };
2 struct B { int u; float v; };
3
4 int foo(int a);
5
6 int bar(struct A p) {
7     return p.x + p.y;
8 }
9
10 int main() {
11     struct A a;
12     struct B b;
13     a.x = 1;
14     a.y = 2;
15     b.u = 3;
16     b.v = 4.0;
17     bar(a);
18     return 0;
19 }
20
21 int bar(struct B q);
```

#### 5.1.2 输出

```
1 Error type 18 at Line 4: Undefined function.
2 Error type 19 at Line 21: Inconsistent declaration.
```

#### 5.1.3 说明

E 类分组 1 专项（函数声明）：分组 0、1 须在第 4 行处报告类型 18；分组 1 还须在第 21 行报告类型 19。

## 5.2 E-2

### 5.2.1 输入

```
1 struct Pt {
2     int x;
3     int y;
4 };
5
6 float area(float r) {
7     if (r > 0.0) {
8         return r;
9     }
10    return 0.0;
11 }
12
13 int main() {
14     float cnt = 0.0;
15     int tot = 0;
16
17     struct Pt p;
18     int buf[3];
19
20     {
21         struct Pt p;
22         int cnt = 0;
23         float tot;
24         float buf[10];
25
26         tot = area(cnt);
27
28         while (cnt > 0) {
29             int buf = 99;
30             cnt = cnt - 1;
31             p.x = cnt;
32             buf[cnt] = 99;
33         }
34     }
```

```

35
36     buf[0] = 99;
37     p.x = 99;
38     return 0;
39 }

```

### 5.2.2 输出

```

1 Error type 9 at Line 26: Function is not applicable for arguments.
2 Error type 10 at Line 32: Not an array.

```

### 5.2.3 说明

E 类分组 2 专项（作用域）：分组 0、2 须在第 26、32 行报告类型 9、10。

## 5.3 E-3

### 5.3.1 输入

```

1 struct S1 { int s1_a; float s1_b; } s1;
2 struct S2 { float s2_d; int s2_c; } s2;
3
4 struct X { float x_arr[3][3]; } x1;
5 struct Y { int y_arr[2][4]; } y1;
6
7 struct P {
8     int p_v[3];
9     int p_n;
10    struct { int p_m_id; float p_m_sc; } p_m;
11 };
12
13 struct Q {
14     int q_n[4];
15     struct { int q_d_id; float q_d_sc; } q_d;
16 };
17
18 struct Q get(struct P p_arg) {
19     return p_arg;

```

```
20 }
21
22 int main() {
23     float t;
24     struct P p_local;
25     struct Q q_local;
26
27     s1 = s2;
28
29     x1 = y1;
30
31     q_local = get(p_local);
32
33     return 0;
34 }
```

### 5.3.2 输出

```
1 Error type 8 at Line 19: Type mismatched for return.
2 Error type 5 at Line 27: Type mismatched for assignment.
3 Error type 5 at Line 29: Type mismatched for assignment.
```

### 5.3.3 说明

E类分组3专项（结构等价）：应在第19行报告类型8错误（返回类型不匹配），第27和29行报告类型5错误（赋值类型不匹配）。

## 6 结束语

若对本文档有任何疑议，可写邮件与陈泰霖助教联系，注意同时抄送给许畅老师。