2017 年春季学期编译原理第二次实验测试用例：目录

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 15,9,7。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

输入

```
struct Product{
    int x;
    int y;
    int z;
}a;
int testFunction(int x1, int y1)
{
    a.x = x1;
    a.y = y1;
    a.z = z1;
    return 0;
}
```

输出

```
Error type 1 at line 10: Undefined variable "z1"
```

说明：a.z = z1 这一句包含未定义的变量 z1，这里也可以另外报出错误类型 5（= 两边类型不匹配）。

1.2 A-2

输入

```
int compare(int m, int n){
    int m1, n1;
```

```
3       if(m < 0)
4           m1 = -m;
5       if(n < 0)
6           n1 = -n;
7       return compareP(m1, n1);
8   }
```

输出

```
1   Error type 2 at line 7: Undefined function "compareP"
```

说明：compareP 函数未定义，也可以爆出连锁反应的错误类型 8。

1.3 A-3

输入

```
1   struct Product{
2       int name;
3       int weight;
4       int price;
5   }p;
6
7   int main(){
8       int Product, x1, x2, x3;
9       p.name = x1;
10      p.weight = x2;
11      p.price = x3;
12      Product = p.weight * p.price;
13      return 0;
14  }
```

输出

```
1   Error type 3 at line 8: Redefined variable "Product"
```

说明：重复定义的变量 Product，这里如果错误位置写为第 1 行也算对，可以在第 12 行报出连锁反应导致的错误类型 1。

1.4 A-4

输入

```
int math_calculator(int a, int b, int c){
    return a + b * c;
}

float math_calculator(float a1, float b1, float c1, float d1){
    return a1 * b1 + c1 * d1;
}

int main()
{
    return math_calculator(5,4,7);
}
```

输出

```
Error type 4 at line 5: Redefined function "math_calculator"
```

说明：重复定义的函数 math_calculator。这里如果没有把重复定义的函数放入符号表，会在第 11 行报了错误类型 2，是否报出这个错误，不影响得分。

1.5 A-5

输入

```
struct Product {
    int name;
    int weight;
    float price;
};
struct ProductB {
    int nameB;
    float weightB;
    float priceB;
```

```
};
int main(){
    struct Product a;
    struct ProductB b;
    a.name = b.nameB;
    a.weight = b.weightB;
    a.price = b.priceB;
    return 0;
}
```

输出

```
Error type 5 at line 15: Type mismatched
```

说明：赋值号两边类型不匹配（float 赋值给 int）。

1.6 A-6

输入

```
int cube (int n){
        return n * n * n;
}

int main()
{
        int a = 100, b = 4;
        if(a < cube(b))
                a = cube(b);
        else
                cube(b) = a;
        return a;
}
```

输出

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

说明：赋值号左边是一个不能为左值的类型（函数）。

1.7 A-7

输入

```
struct List{
        int name;
        int list[10];
};
int main(){
        struct List a, b, c;
        int i, N =10;
        while(i < N){
                a.list[i] = b.list+c.list[i];
                i = i + 1;
        }
        return 0;
}
```

输出

```
Error type 7 at line 9: Operands type mismatched
```

说明：加号操作符两边类型不匹配，这里可以另外报错误类型 5（赋值号两边错误类型不匹配），必须在 9 行。

1.8 A-8

输入

```
struct Vector{
        float x;
        float y;
        float list[5];
};
```

```
6  struct Vector structMutipleFunction(struct Vector A, struct Vector B)
       {
7          struct Vector result;
8          int i, N =5;
9          result.x = A.x + B.x;
10         result.y = A.y + B.y;
11         while(i < N){
12                 result.list[i] = A.list[i] + B.list[i];
13         }
14         return result.list;
15 }
```

输出

```
1  Error type 8 at line 14: The return type mismatched
```

说明：返回值实际类型与函数定义不一致，报在第 6 行也是对的。

1.9 A-9

输入

```
1  struct INTPair{
2          int a;
3          int b;
4  };
5  struct INT{
6          int a2;
7          int b2;
8          int c2;
9  }q;
10 int split(struct INTPair pair)
11 {
12         int first = pair.a;
13         int last = pair.b;
14         int x[10];
```

```
15          int pivot = first;
16          int split_point = first;
17          int i = first + 1;
18          int temp,temp2;
19          while(i <= last)
20          {
21                  if(x[i] < x[pivot])
22                  {
23                          split_point = split_point + 1;
24                          temp = x[i];
25                          x[i] = x[split_point];
26                          x[split_point] = temp;
27                  }
28                  i = i + 1;
29          }
30          temp2 = x[pivot];
31          x[pivot] = x[split_point];
32          x[split_point] = temp2;
33          return split_point;
34  }
35
36  int main()
37  {
38          return split(q);
39  }
```

输出

```
1  Error type 9 at line 38: The method "split" is not applicable for the
       arguments
```

说明：函数实参与形参类型不一致。

1.10 A-10

输入

```
int main()
{
        int a[10][10], b[10][10];
        int i, j, N = 10;
        while(i < N)
        {
                while(j < N)
                {
                        a[i][j] = b[i][j] + i * j;
                        a[i][j-1] = a[i][j][0];
                        j = j + 1;
                }
                i = i + 1;
        }
        return 0;
}
```

输出

```
Error type 10 at line 10: Illegal use of "[]"
```

说明：对非数组变量使用 [] 操作符，这里会连带报出错误类型 5，因为赋值号右边的类型可以算作是“未知”。

1.11 A-11

输入

```
int bbSort(int n)
{
        int exchange;
        int i, j, temp;
        int p[50];
```

```
        while(i < n)
        {
                i = i + 1;
                exchange = 1;

                while(j < n)
                {
                        j = j + 1;
                        if(p[j] > p[j+1])
                        {
                                temp = p[j+1];
                                p[j+1] = p[j];
                                p[j] = temp;
                                exchange = 0;
                        }
                }
                if(exchange() == 0)
                        p[i] = -1;
        }
        return 0;
}
```

输出

```
Error type 11 at line 22: "exchange" must be a function
```

说明：对非函数的标识符使用 () 操作符。

1.12 A-12

输入

```
int main()
{
        int a[10];
        int i = 0, N = 10;
```

```
5           int max = 0;
6           while (i < N){
7                   a[i] = i * i - i * 2 + 1;
8                   i = i + 1;
9           }
10          i = 0;
11          while (i < N){
12                  if (max < a[i]){
13                          max = a[0.5];
14                          i = i + 1;
15                  }
16          }
17          return 0;
18  }
```

输出

```
1  Error type 12 at line 13: Array argument is not an integer
```

说明：数组下标非整数。

1.13 A-13

输入

```
1  struct Type{
2           int name;
3           int operator;
4  };
5  struct {
6           struct Type a;
7           int b;
8  } v;
9  int test_function()
10 {
11     int temp = 2;
```

```
        v.a.name = temp + v.b;
        v.a.operator = temp;
        v.b.name = v.b * v.b;
        return v.b;
}
```

输出

```
Error type 13 at line 14: Illegal use of "."
```

说明：对非结构体变量使用“.”操作符，同时可以报出错误类型 5。

1.14 A-14

输入

```
struct _price {
        float a, b;
};

struct _product {
    float x, y;
        struct _price price;
};

int main()
{
    struct _product v1, v2;
        float p,q;
        p = v1.x + v1.y * v1.price.a;
        q = v2.x + v2.y * v2.b * v2.price.b;
        return 0;
}
```

输出

```
Error type 14 at line 15: Un-existed field "b"
```

说明：使用了结构体中未定义的域 b，这里可以报出错误类型 5 和 7。

1.15 A-15

输入

```
struct Food {
    int num, weight;
        float price;
        float time;
};

int main()
{
        struct Food food;
        struct Food2{
                int weight1 = 2;
        }food2;
        food.weight = food2.weight1;
        return food.weight;
}
```

输出

```
Error type 15 at line 11: Initialized struct field
```

说明：结构体内部域进行初始化。有的同学由于 Food2 定义错误，就没有将其放入符号表，因此会在第 13 行报 food2 未定义等连锁错误，这个不影响得分。

1.16 A-16

输入

```
struct Product{
        int name;
        int weight;
        struct Price{
                int d;
```

```
6                   int p;
7           } price;
8   }product;
9   struct Price{
10          int p1;
11          int p2;
12  };
13
14  int main(){
15          product.name = 1;
16          product.weight = 2;
17          return product.name * product.weight;
18  }
```

输出

```
1   Error type 16 at line 9: Duplicated name "Price"
```

说明：重复定义的结构体 Price，也可以报错在第 4 行。

1.17 A-17

输入

```
1   struct Food {
2       int price, weight;
3   };
4   struct Drink {
5           int sweet1;
6           int price1;
7   };
8   struct Drink2 {
9           int sweet2;
10          int price2;
11  };
12  int main()
```

```
{
        struct Food meat;
        struct Food2 vegetable;
        struct Drink orange;
    return 0;
}
```

输出

```
Error type 17 at line 15: Undefined struct "Food2"
```

说明：使用了未定义的结构体 Food2。

1.18 A-18

输入

```
struct Food {
        int price;
        float weight;
        struct Price{
                int p;
        }price;
};

int main()
{
        struct Food food;
        food.price = 25;
        food.weight = 0.5;
        return 0;
}
```

输出

```
Error type 15 at line 6: Redefined field "price"
```

说明：在结构体中重复定义域。

1.19 A-19

输入

```
struct Food {
        int price;
        float weight;
}food;

int set_Item(struct Food temp){
        temp.price = 10;
        temp.weight = 0.5;
        return 0;
}
int main()
{
        int temp1 = 20;
    struct Food food2;
    food2.price = temp1;
    food2.weight = 0.5;
        set_Item(food, food2);
        return 0;
}
```

输出

```
Error type 9 at line 17: The method "set_Item" is not applicable for
   the arguments
```

说明：函数实参与形参数目不一致。

1.20 A-20

输入

```
struct Food {
        int price;
```

```
        float weight;
}food;

int main()
{
        int a[10];
        int temp = 20;
    struct Food food2;
    food2.price = 25;
    food2.weight = 0.5;
        food.price = temp;
        food = a + food2;
        return 0;
}
```

输出

```
Error type 7 at line 14: Type mismatched
```

说明：操作符与操作数不匹配，多报错误类型 5 也可以。

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。每一行的语义错误会分别算分，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

输入

```
struct Product{
        int name;
        float weight;
        float price;
        int date_from;
```

```
6           int date_end;
7   };
8   struct salesList{
9           float sale_no[100];
10          int date;
11  };
12
13  struct Product set_ProductSingle(int name1, float weight1, float
        price1, int date_from1, int date_end1){
14          struct Product result;
15          result.name = name1;
16          result.weight = weight1;
17          result.price = price1;
18          result.date_frm = date_from1;
19          result.date_end = date_end1;
20          return result;
21  }
22
23  struct saleList set_SalesList(float sale_no1[100], int date1){
24          struct salesList result2;
25          int i, N = 100;
26          while(i < N){
27                  result2.sale_no[i] = sale_no1[i];
28                  i = i + 1;
29          }
30          result2.date = date1;
31          return result2;
32  }
33
34  float calculateAll(struct Product p, struct salesList s){
35          int resultAll;
36          float sum;
```

```
37          int i1, N1 = 100;
38          while(i1 < N1){
39                  sum = s.sale_no[i1] + sum;
40                  i1 = i1 + 1;
41          }
42          return sum * p.price * p.weight;
43  }
44
45  int main(){
46          struct Product p1;
47          struct salesList s1;
48          float list[100];
49          calculateAll(set_ProductSingle(1,2.0,3.0,4.0,5),set_SalesList
                (list,6));
50          calculateAll(p1,s1,1);
51          return 0;
52  }
```

输出

```
1  Error type 17 at Line 23: Undefined structure "saleList"
2  Error type 14 at Line 18: Non-existence field "date_frm"
3  Error type 9 at Line 49: Function "set_ProductSingle" is not
       applicable for these arguments
4  Error type 9 at Line 50: Function "calculateAll" is not applicable
       for these arguments
```

说明：输出中的 4 个错误为本质错误，是必须要报出来的，这些错误可能会有连锁反应，例如：31 行可以多报返回值类型不匹配的错误，以及 49 行可以多报 calculateAll 函数的参数调用不合理的错误，连锁反应不限于此，合理即可。

2.2 B-2

输入

```
1  struct {
```

```
        int price;
        int sales[100];
} salesDay[5];

struct Sum {
        int price_sum = 0;
        int sales_sum;
};

struct Summ MethodDisplay(){
        struct Sum sum_result;
        int i, j, sum_all, sum, M = 5, N = 100;
        int price_all = 0;
        while(i < M){
                sum = 0;
                while(j < N){
                        sum = sum + salesDay[i].sales[j];
                        j = j[i] + 1;
                }
                i = i + 1;
                sum_all = sum_all + sum;
                price_all = salesDay[i].price * sum + price_all;
        }
        sum_result.price_sum = price_all;
        sum_result1.sales_sum = sum_all;
        return sum_result;
}
```

输出

```
Error type 15 at Line 7: Initialized variable "price_sum" in struct
Error type 17 at Line 11: Undefined structure "Summ"
Error type 10 at Line 19: "j" is not an array
Error type 1 at Line 26: Undefined variable "sum_result1"
```

说明：输出中的 4 个错误为本质错误，是必须要报出来的，这些错误可能会有连锁反应，例如：第 19 行的错误可能会导致错误类型 5 和 7，因为 j[i] 的类型未知；第 26 行可能会报出错误类型 13，因为这里 sum_result1 类型未知，所以符号存在错误；第 27 行可能会报出一个类型 8 错误，连锁不仅限与此，合理即可。

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

输入

```
1  struct Product{
2          int name;
3          float weight;
4          float price;
5          int date_from;
6          int date_end;
7  };
8  struct salesList{
9          float sale_no[100];
10         int date;
11 };
12
13 struct Product set_ProductSingle(int name1, float weight1, float
       price1, int date_from1, int date_end1){
14         struct Product result;
15         result.name = name1;
16         result.weight = weight1;
17         result.price = price1;
18         result.date_from = date_from1;
19         result.date_end = date_end1;
20         return result;
```

```
21 }
22
23 struct salesList set_SalesList(float sale_no1[100], int date1){
24         struct salesList result2;
25         int i, N = 100;
26         while(i < N){
27                 result2.sale_no[i] = sale_no1[i];
28                 i = i + 1;
29         }
30         result2.date = date1;
31         return result2;
32 }
33
34 float calculateAll(struct Product p, struct salesList s){
35         int resultAll;
36         float sum;
37         int i1, N1 = 100;
38         while(i1 < N1){
39                 sum = s.sale_no[i1] + sum;
40                 i1 = i1 + 1;
41         }
42         return sum * p.price * p.weight;
43 }
44
45 int main(){
46         struct Product p1;
47         struct salesList s1;
48         float list[100];
49         calculateAll(set_ProductSingle(1,2.0,3.0,4,5),set_SalesList(
              list,6));
50         calculateAll(p1,s1);
51         return 0;
```

```
}
```

输出

```
//正常返回，没有任何输出。
```

说明：本测试用例是 B_1 类测试用例的改正版。

3.2 C-2

输入

```
struct {
        int price;
        int sales[100];
} salesDay[5];

struct Sum {
        int price_sum;
        int sales_sum;
};

struct Sum MethodDisplay(){
        struct Sum sum_result;
        int i, j, sum_all, sum, M = 5, N = 100;
        int price_all = 0;
        while(i < M){
                sum = 0;
                while(j < N){
                        sum = sum + salesDay[i].sales[j];
                        j = j + 1;
                }
                i = i + 1;
                sum_all = sum_all + sum;
                price_all = salesDay[i].price * sum + price_all;
        }
```

```
25          sum_result.price_sum = price_all;
26          sum_result.sales_sum = sum_all;
27          return sum_result;
28  }
```

输出

```
1  //正常返回，没有任何输出。
```

说明：本测试用例是 B_2 类测试用例的改正版。

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

输入

```
1  struct Circle{
2          int x;
3          int y;
4          int r;
5  };
6
7  struct Square{
8          int a;
9          int b;
10 };
11
12 int Calculation1(struct Square square){
13         return square.a * square.b;
14 }
15
16 int Calculation2(struct Circle circle1);
```

```
17 int Calculation2(struct Circle circle){
18         return 3 * circle.r * circle.r;
19 }
20
21 int Compare(struct Square s, struct Circle c);
22 int Compare(struct Square s, struct Circle c){
23         if(Calculation1(s)>Calculation2(c))
24                 return 1;
25         else return -1;
26 }
```

输出

```
1 //正常返回，没有任何输出。
```

说明：对于 2.1 分组的同学，应该没有任何输出，对于其他分组的同学，应该在第 16,21 行报出有语法错误 (Error type B)。

4.2 D-2

输入

```
1  struct Node{
2          int no;
3          int array[5];
4          float value;
5          int nextno;
6  }node[100];
7
8  int initial_Node(struct Node temp1){
9          int i = 0;
10         while(i < 5){
11                 temp1.array[i] = i + i * (2 * i + 1);
12                 i = i + 1;
13         }
14         temp1.value = 0.0;
```

```
15          temp1.no = temp1.nextno = -1;
16          return 1;
17  }
18
19  int link_Node(struct Node former, struct Node later){
20          int result = 0;
21          if(former.nextno!=-1 && later.nextno == -1){
22                  int result = 0;
23                  later.nextno = former.nextno;
24                  former.nextno = later.no;
25                  return result;
26          }
27          else if(former.nextno == -1 && later.nextno != -1){
28                  int result = 1;
29                  former.nextno = later.no;
30                  return result;
31          }
32          return -1;
33  }
34
35  int main(){
36          struct Node a, b;
37          int i, N = 100;
38          initial_Node(a);
39          initial_Node(b);
40          link_Node(a,b);
41          while(i < N){
42                  initial_Node(node[i]);
43                  i = i + 1;
44          }
45          i = 0;
46          while(i < N){
```

```
47                  link_Node(node[i],node[i+1]);
48                  i = i + 1;
49          }
50          return 0;
51
52  }
```

输出

```
1  //正常返回，没有任何输出。
```

说明：对于 2.2 分组的同学，应该没有任何输出，其他分组的同学应该会识别出大量的重复定义变量（result 和 i)。

4.3 D-3

输入

```
1  struct A{
2          int a;
3          int a_array[10][5];
4          struct innerA{
5                  int innera[10];
6          } innerA_node;
7  };
8
9  struct B{
10         int b;
11         int b_array[5][2];
12         struct innerB{
13                 int innerb[20];
14         } innerB_node;
15 };
16
17 struct C{
18         int c;
```

```
19          float cc;
20  };
21
22  struct D{
23          int d;
24          float dd;
25  };
26
27  int main(){
28          struct A tempA, tempA2;
29          struct B tempB;
30          struct C tempC, tempC2;
31          struct D tempD;
32          tempA = tempB;
33          tempC = tempD;
34          tempA = tempA2;
35          tempC = tempC2;
36          return 0;
37  }
```

输出

```
1  //正常返回，没有任何输出。
```

说明：对于分组 2.3 的同学，应该没有任何输出，其他分组的同学应该在 32 行 33 行识别出类型不匹配。（函数参数类型 Error type 5）

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

这组测试用例针对 2.1 分组的同学

输入

```
struct Circle{
        int x;
        int y;
        int r;
};

struct Square{
        int a;
        int b;
};

int Calculation1(struct Square square){
        return square.a * square.b;
}

int Calculation2(struct Circle circle1);
int Calculation2(struct Circle circle){
        return 3 * circle.r * circle.r;
}

int Calculation3(struct Circle circle3);
int Calculation3(struct Circle circle3,struct Circle circle33);

int Compare(struct Square s, struct Circle c){
        if(Calculation1(s)>Calculation2(c))
                return 1;
        else return -1;
}
int Compare(struct Square s, struct Circle c,struct Circle cc);
```

输出

```
Error type 19 at line 22: Inconsistent declaration of function "
```

```
   Calculation3"
Error type 19 at line 29: Inconsistent declaration of function "
   Compare"
Error type 18 at line 21: Undefined function "Calculation3"
```

说明：仅 2.1 分组同学需要测试该用例，需要输出上述的错误信息，其中错误类型 18 也可以输出在第 22 行，错误类型 19 也可以输出在第 21,24 行。

5.2 E-2

这组测试用例针对 2.2 分组的同学

输入

```
struct Node{
        int no;
        int array[5];
        float value;
        int nextno;
}node[100];

int initial_Node(struct Node temp1){
        int i = 0;
        while(i < 5){
                temp1.array[i] = i + i * (2 * i + 1);
                i = i + 1;
        }
        temp1.value = 0.0;
        temp1.no = temp1.nextno = -1;
        return 1;
}

int link_Node(struct Node former, struct Node later){
        if(former.nextno!=-1 && later.nextno == -1){
                int result = 0;
                later.nextno = former.nextno;
```

```
                former.nextno = later.no;
                return result;
        }
        else if(former.nextno == -1 && later.nextno != -1){
                result = 1;
                former.nextno = later.no;
                return result;
        }
        return -1;
}

int main(){
        struct Node a, b;
        int i, a, N = 100;
        initial_Node(a);
        initial_Node(b);
        link_Node(a,b);
        while(i < N){
                initial_Node(node[i]);
                i = i + 1;
        }
        i = 0;
        while(i < N){
                link_Node(node[i],node[i+1]);
                i = i + 1;
        }
        return 0;

}
```

输出

```
Error type 1 at Line 27: Undefined variable "result"
Error type 3 at Line 36: Redefined variable "a"
```

说明：仅 2.2 分组同学需要测试该用例，需要输出上述的错误信息，可能会导致 27 和 29 行出现连锁反应，合理即可。

5.3 E-3

这组测试用例针对 2.3 分组的同学

输入

```
struct A{
        int a;
        int a_array[10][5];
        struct innerA{
                int innera[10];
        } innerA_node[2];
};

struct B{
        int b;
        int b_array[5][2];
        struct innerB{
                float innerb[20];
        } innerB_node[4];
};

struct C{
        int c;
        float cc[10][2];
};

struct D{
        int d;
        float dd[3][4];
};
```

```
26
27 struct E{
28         int e;
29         int e_array[5][2];
30         struct innerE{
31                 float innere[25][5];
32         } innerE_node[6];
33 };
34
35 int main(){
36         struct A tempA;
37         struct B tempB;
38         struct C tempC;
39         struct D tempD;
40         struct E tempE;
41         tempA = tempB;
42         tempC = tempD;
43         tempB = tempE;
44         return 0;
45 }
```

输出

```
1 Error type 5 at line 41: Type mismatched for assignment
2 Error type 5 at line 43: Type mismatched for assignment
```

说明：仅 2.3 分组同学需要测试该用例，需要输出上述的错误信息。

6 结束语

如果对本测试用例有任何疑议，可以写邮件与王慧妍助教联系，注意同时抄送给许老师。