

2017 年春季学期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	4
1.4	A-4	5
1.5	A-5	6
2	B 组测试用例	7
2.1	B-1	7
2.2	B-2	7
2.3	B-3	9
3	C 组测试用例	10
3.1	C-1	10
3.2	C-2	12
4	D 组测试用例	14
5	E 组测试用例	15
5.1	E1-1	15
5.2	E1-2	16
5.3	E1-3	17
5.4	E2-1	18
5.5	E2-2	20
5.6	E2-3	21
6	结束语	24

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 {
3     int a, b, c;
4     int final = 0;
5     a = 5;
6     b = a * a * (a + 2);
7     write(b);
8     c = b / a + 1;
9     write(c);
10    final = a + b - c * 3 + (b / a - 4);
11    write(final);
12    return 0;
13 }
```

程序输入：无；预期输出：175 36 103

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 {
3     int month, year;
4     year = read();
5     month = read();
6     if(year == (year / 100 * 100)){
```

```

7      if(year == (year / 400 * 400)){
8          if(month == 2){
9              write(29);
10         }
11         else if(month == (month / 2 * 2)){
12             write(30);
13         }
14         else
15             write(31);
16     }else {
17         if(month == 2){
18             write(28);
19         }
20         else if(month == (month / 2 * 2)){
21             write(30);
22         }
23         else {
24             write(31);
25         }
26     }
27 }
28 else if(year == (year/4 * 4)){
29     if(month == 2){
30         write(29);
31     }
32     else if(month == (month / 2 * 2)){
33         write(30);
34     }
35     else {
36         write(31);
37     }
38 }

```

```

39     else{
40         if(month == 2){
41             write(28);
42         }
43         else if(month == (month / 2 * 2)){
44             write(30);
45         }
46         else {
47             write(31);
48         }
49     }
50     return 0;
51 }

```

输入：1998 3；输出：31

输入：1999 10；输出：30

输入：2000 2；输出：29

输入：2100 2；输出：28

说明：这个测试用例主要针对分支语句进行测试，是一个输入年与月输出当月所包含的天数的程序。注意，程序每次输入以空格隔开表示，每次输入一个数（下同）。

1.3 A-3

输入

```

1  int main(){
2      int N = 30;
3      int num = 0, i = 1, k = 1;
4      while(k <= N){
5          while(i <= k){
6              if(k == (k / i * i))
7                  num = num + 1;
8              i = i + 1;
9          }
10         if(num == 2)

```

```

11         write(k);
12         i = 1;
13         num = 0;
14         k = k + 1;
15     }
16     return 0;
17 }

```

输入：无；输出：2 3 5 7 11 13 17 19 23 29

说明：这个测试用例依次输出 30 以内的所有素数。

1.4 A-4

输入

```

1  int main(){
2      int x[5], max, temp;
3      int i = 0, N = 5, j = 0;
4      while(i < 5){
5          x[i] = read();
6          i = i + 1;
7      }
8      i = N;
9      while(i > 0){
10         while(j < i - 1){
11             if(x[j] > x[j + 1]){
12                 temp = x[j];
13                 x[j] = x[j + 1];
14                 x[j + 1] = temp;
15             }
16             j = j + 1;
17         }
18         j = 0;
19         i = i - 1;
20     }

```

```

21     i = 0;
22     while(i < 5){
23         write(x[i]);
24         i = i + 1;
25     }
26     return 0;
27 }

```

输入: 7 6 5 3 4; 输出:3 4 5 6 7

说明: 这个测试用例是一个冒泡排序的小例子。

1.5 A-5

输入

```

1  int fact(int m){
2      int result = 1;
3      int j = 1;
4      while(j <= m){
5          result = result * j;
6          j = j + 1;
7      }
8      return result;
9  }
10 int main(){
11     int x[5];
12     int i = 0, N = 5;
13     while(i < N){
14         x[i] = i + 2;
15         write(fact(x[i]));
16         i = i + 1;
17     }
18     return 0;
19 }

```

输入：无；输出：2 6 24 120 720

说明：这个测试用例输出 2 3 4 5 6 五个数的阶乘结果，主要测试函数的简单调用。

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

输入

```
1 int hanoi(int n, int p1, int p2, int p3){
2     if(n == 1){
3         write(p1*1000000+p3);
4     }
5     else{
6         hanoi(n-1,p1,p3,p2);
7         write(p1*1000000+p3);
8         hanoi(n-1,p2,p1,p3);
9     }
10    return 0;
11 }
12 int main(){
13     int sum = 3;
14     hanoi(sum, 1, 2, 3);
15     return 0;
16 }
```

输入：无；输出：1000003 1000002 3000002 1000003 2000001 2000003 1000003

说明：这个测试用例是一个简单的三层汉诺塔问题，其中输出 1000003 表示将杆 1 最上面的一个木块移到杆 3，主要考察关于递归函数的翻译。

2.2 B-2

输入

```

1  int gcd1(int a, int b)
2  {
3      int result;
4      int temp;
5      temp = a - a / b * b;
6      while (temp != 0) {
7          a = b;
8          b = temp;
9          temp = a - a / b * b;
10     }
11     result = b;
12     return result;
13 }
14 int gcd2(int x, int y)
15 {
16     int remainder = 0;
17     if(x > y){
18         remainder = x - x / y * y;
19         if(remainder == 0)
20             return y;
21         else
22             return gcd2(y, remainder);
23     }
24     else {
25         remainder = y - y / x * x;
26         if(remainder == 0)
27             return x;
28         else
29             return gcd2(x, remainder);
30     }
31 }
32

```



```

33 int main()
34 {
35     int m = read();
36     int n = read();
37     write(gcd1(m,n));
38     write(gcd2(m,n));
39     return 0;
40 }

```

输入：100 96；输出：4 4 输入：20 100；输出：20 20

说明：这个测试用例是计算最大公约数的小程序，两种计算方法可选。

2.3 B-3

输入

```

1 int sqaRever(int num)
2 {
3     int flag = 0;
4     int array[3];
5     int j = 0;
6     array[0] = num / 100;
7     array[1] = num / 10 - 10 * array[0];
8     array[2] = num - 100 * array[0] - 10 * array[1];
9     if (array[0] != array[2]) {
10         flag = 0;
11     }
12     else {
13         while (j < 12){
14             if ((j * j - num) == 0 )
15                 flag = 1;
16             j = j + 1;
17         }
18     }
19     if (flag == 1)

```

```

20         return 1;
21     else
22         return 0;
23 }
24
25 int main()
26 {
27     int i = 100;
28     while (i < 150){
29         if (squaRever(i) == 1)
30             write(i);
31         i = i + 1;
32     }
33     return 0;
34 }

```

输入：无；输出：121

说明：这个测试用例是计算平方回文数的程序，输出 100 到 150 之间的平方回文数。

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 mod(int number2, int m2)
2 {
3     int result = number2 - number2 / m2 * m2;
4     int result2 = result;
5     return result;
6 }
7
8 int power(int base1, int p1) {

```

```

9      int ret1 = 1 + p1 - p1;
10     while(p1 > (ret1 - ret1 + 90 - 89 + 1 - 2))
11     {
12         ret1 = ret1 * base1;
13         p1 = 2 * 1 * p1 - 1 * p1 - 1;
14     }
15     return ret1;
16 }
17
18 int getNumDigits(int number3)
19 {
20     int ret3 = 0;
21     if(number3 < 0)
22         return -1;
23     while(number3 > 0) {
24
25         number3 = number3 / 10;
26         ret3 = ret3 + 2;
27         ret3 = ret3 + 2;
28         ret3 = ret3 - 3;
29     }
30
31     return ret3;
32 }
33
34 int isNarcissistic(int number4)
35 {
36     int numDigits4 = getNumDigits(1 + number4 - 1);
37     int sum4 = 0;
38     int n4 = number4;
39     int s4;
40     while(n4>0) {

```

```

41     s4 = mod(n4, 10);
42     n4 = (n4 - s4) / 10;
43     sum4 = sum4 + power(s4, numDigits4);
44 }
45
46 if(sum4 == number4)
47     return 1;
48 else
49     return 0;
50 }
51
52 int main() {
53     int count = 0;
54     int i = 300;
55     while(i < 500)
56     {
57         if(isNarcissistic(i) == 1)
58         {
59             write(i);
60             count = count + 1;
61         }
62         i = i + 1;
63     }
64     write(count);
65     return count;
66 }

```

输入：无；输出：370 371 407 3

说明：这个测试用例计算 300 到 500 之间的水仙花数，并输出满足条件的数的个数。

3.2 C-2

输入

```

1 int Josepy(int n, int m) {

```

```

2   int people[100], i = 1;
3   int j = 0, k = 0, num = 0;
4   while(i <= n){
5       people[i] = 1;
6       i = i + 1;
7   }
8   i = 1;
9   while(i <= n){
10      if(people[i] == 1){
11          j = j + people[i];
12          if(j == m){
13              write(i);
14              j = 0;
15              people[i] = 0;
16              k = k + 1;
17          }
18          if(k == n){
19              num = i;
20              return 0;
21          }
22      }
23      if(i == n)
24          i = 0;
25      i = i + 1;
26  }
27  return 0;
28  }
29  int main(){
30      int a, b;
31      a = read();
32      b = read();
33      Josepy(a, b);

```

```
34     return 0;
35 }
```

输入：10 3； 输出：3 6 9 2 7 1 8 5 10 4

说明：这个测试用例实现了一个经典的约瑟夫问题。

4 D 组测试用例

本组测试用例 1 个，测试用例主要用于测试中间代码的优化。

```
1  int changeCal(int r1)
2  {
3      int r2;
4      r2 = r1 + 345 - 345 + r1 * r2 - r1 * r2 + r1 * r1 - r1 * r1;
5      r2 = r2 * 2 - r2 * 3 + r2 * 2 + 34 / 2 + 10 / 3 - 20 / 1;
6      r2 = 7 + 6 - 11 * 1;
7      r2 = r1 + 4 * 5 - 10 * 2 - 5 / 6;
8      return r2;
9  }
10
11 int main()
12 {
13     int a = 7 + 6 - 11 * 1, b = 8 - 4, c = 4 + 4 + 5 * 6 / 3 - 4;
14     int d = a + b - c;
15     int e = a + b + c * 2;
16     int f = a + b + c;
17     int g1 = 42, i = 0;
18     int g, h;
19     f = a + b + c - d - e + f;
20     while (b - a < f) {
21         g1 = g1 + i * 4 + 3 + 4 + 5;
22         g = f - changeCal(b) + a * 2 + c * d - f;
23         i = i + 1 + 0;
24         i = i + 3 + 1;
```

```

25         i = i - 2 - 2;
26         if (i - i / 3 * 3 == a - changeCal(a) + b - b + c - c)
27             f = f + 1 + 1;
28         f = f - 2 + 1;
29     }
30     h = g1 - 2 + 3;
31     write(h);
32     i = g1;
33     while (g1 >= 1200 + 22) {
34         i = g1 + 1024;
35         g1 = changeCal(g1) - 1;
36         i = g1;
37     }
38     write(g1);
39     a = a + b;
40     b = a + b;
41     c = a + b;
42     write(c);
43     return 0;
44 }

```

输入：无；输出：2411 1221 16

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 weight;
3     int price;
4 };
5
6 int main()
7 {
8     struct Product p1;
9     p1.weight = 123;
10    p1.price = 56;
11    write(p1.weight * p1.price);
12    return 0;
13 }

```

输入：无；输出：6888

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 Worker{
2     int salary;
3     int year;
4     int esca;
5 };
6
7 int main(){
8     struct Worker worker[10];
9     int i, j, add, N = 10;
10    int final = 0;
11    while(i < N){
12        worker[i].salary = 100;

```



```

13     worker[i].year = i;
14     worker[i].esca = i * i - i + 2;
15     i = i + 1;
16 }
17 while(j < N){
18     if(worker[j].year < 3)
19         add = 1;
20     else
21         add = 2;
22     final = final + worker[j].salary * (365 - worker[j].esca) *
23         add;
24     j = j + 1;
25 }
26 write(final);
27 return 0;
28 }

```

输入：无；输出：569300

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 Detail{
2     int score;
3     int name;
4 };
5
6 struct ScoreClass{
7     struct Detail scoreDetail[100];
8     int num;
9 };
10

```

```

11 int average(struct ScoreClass class){
12     int scoreSum = 0;
13     int i, N = class.num;
14     while(i < N){
15         scoreSum = class.scoreDetail[i].score + scoreSum;
16         i = i + 1;
17     }
18     return scoreSum / N;
19 }
20
21 int main(){
22     struct ScoreClass classInit;
23     int result = 0, initN = 10;
24     int j = 0;
25     classInit.num = initN;
26     while(j < initN){
27         classInit.scoreDetail[j].score = (j + 1) * (j + 1);
28         j = j + 1;
29     }
30     result = average(classInit);
31     write(result);
32     return 0;
33 }

```

输入：无；输出：38

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 array[5][4][3];
3     int r = 5, c = 4, d = 3;

```

```

4      int i = 0, j = 0, k = 0, sum = 0;
5      while(i < r){
6          j = 0;
7          while(j < c){
8              k = 0;
9              while(k < d){
10                 array[i][j][k] = (i + 1) * (j + 2) * (k + 3) + 4;
11                 k = k + 1;
12             }
13             j = j + 1;
14         }
15         i = i + 1;
16     }
17     i = 0;
18     j = 0;
19     k = 0;
20     while(i < r){
21         j = 0;
22         while(j < c){
23             k = 0;
24             while(k < d){
25                 sum = array[i][j][k] + sum;
26                 k = k + 1;
27             }
28             j = j + 1;
29         }
30         i = i + 1;
31     }
32     write(sum);
33
34     return 0;
35 }

```

输入：无；输出：2760

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 print(int array[3], int len){
2     int i1 = 0;
3     while(i1 < len){
4         write(array[i1]);
5         i1 = i1 + 1;
6     }
7     return 0;
8 }
9
10 int InnerP(int array1[3], int array2[3]){
11     int result = 0;
12     int i = 0, N = 3;
13     while(i < N){
14         result = result + array1[i] * array2[i];
15         i = i + 1;
16     }
17     return result;
18 }
19
20 int main(){
21     int p[3], q[3];
22     int k = 0, M = 3;
23     while(k < M){
24         p[k] = k * k;
25         q[k] = 2 * k + 1;
26         k = k + 1;
```

```

27     }
28     print(p, 3);
29     print(q, 3);
30     write(InnerP(p,q));
31
32     return 0;
33 }

```

输入：无；输出：0 1 4 1 3 5 23

说明：测试对于数组作为函数参数的翻译，是一个对一位数组表示的向量求内积的程序。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 chess2[16]){
2
3      int chessplate[4][4];
4      int i2 = 0, j2 = 0;
5      while(i2 < 4){
6          j2 = 0;
7          while(j2 < 4){
8              chessplate[i2][j2] = chess2[i2 * 4 + j2];
9              j2 = j2 + 1;
10         }
11         i2 = i2 + 1;
12     }
13     i2 = 0;
14     j2 = 0;
15     while(i2 < 4){
16         j2 = 0;
17         while(j2 < 4){
18             if(chessplate[i2][j2] == 1)
19                 write(j2);

```

```

20         j2 = j2 + 1;
21     }
22     i2 = i2 + 1;
23 }
24 return 0;
25 }
26
27 int PutQueen(int chess[16], int a[4], int b[7], int c[7], int n, int
sum)
28 {
29     int col, i, j;
30     col = 0;
31     while(col < 4)
32     {
33         if(a[col] && b[n + col] && c[n - col + 3])
34         {
35             chess[n * 4 + col] = 1;
36             a[col] = 0;
37             b[n + col] = 0;
38             c[n - col + 3] = 0;
39             if(n == 3)
40             {
41                 sum = sum + 1;
42                 display(chess);
43             }
44             else
45                 sum = PutQueen(chess, a, b, c, n+1, sum);
46
47             chess[n * 4 + col]=0;
48             b[n + col]=1;
49             c[n - col + 3]=1;
50             a[col]=1;

```

```

51         }
52         col = col + 1;
53     }
54     return sum;
55 }
56
57 int main()
58 {
59     int chess1[16];
60     int a1[4], b1[7], c1[7];
61     int sum1 = 0;
62     int i1;
63     i1 = 0;
64     while(i1 < 16){
65         chess1[i1] = 0;
66         i1 = i1 + 1;
67     }
68     i1 = 0;
69     while(i1 < 4){
70         a1[i1] = 1;
71         i1 = i1 + 1;
72     }
73     i1 = 0;
74     while(i1 < 7){
75         b1[i1] = 1;
76         c1[i1] = 1;
77         i1 = i1 + 1;
78     }
79     sum1 = PutQueen(chess1, a1, b1, c1, 0, sum1);
80     write(sum1);
81
82     return 0;

```

输入：无；输出：1 3 0 2 2 0 3 1 2

说明：测试对于较复杂的数组操作的翻译，是一个八皇后简化后的四皇后问题程序，输出所有满足条件的放置方法和总的解法数目。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，可以写邮件与王慧妍助教联系，注意同时抄送给许老师。