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

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

```
int main()
{
    int a, b, c;
    int final = 0;
    a = 5;
    b = a * a * (a + 2);
    write(b);
    c = b / a + 1;
    write(c);
    final = a + b - c * 3 + (b / a - 4);
    write(final);
    return 0;
}
```

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

```
int main()
{
    int month, year;
    year = read();
    month = read();
    if(year == (year / 100 * 100)){
```

```
        if(year == (year / 400 * 400)){
            if(month == 2){
                write(29);
            }
            else if(month == (month / 2 * 2)){
                write(30);
            }
            else
                write(31);
        }else {
            if(month == 2){
                write(28);
            }
            else if(month == (month / 2 * 2)){
                write(30);
            }
            else {
                write(31);
            }
        }
    }
    else if(year == (year/4 * 4)){
        if(month == 2){
            write(29);
        }
        else if(month == (month / 2 * 2)){
            write(30);
        }
        else {
            write(31);
        }
    }
```

```
    else{
        if(month == 2){
            write(28);
        }
        else if(month == (month / 2 * 2)){
            write(30);
        }
        else {
            write(31);
        }
    }
    return 0;
}
```

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

```
int main(){
    int N = 30;
    int num = 0, i = 1, k = 1;
    while(k <= N){
        while(i <= k){
            if(k == (k / i * i))
                num = num + 1;
            i = i + 1;
        }
        if(num == 2)
```

```
            write(k);
        i = 1;
        num = 0;
        k = k + 1;
    }
    return 0;
}
```

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

```
int main(){
    int x[5], max, temp;
    int i = 0, N = 5, j = 0;
    while(i < 5){
        x[i] = read();
        i = i + 1;
    }
    i = N;
    while(i > 0){
        while(j < i -1){
            if(x[j] > x[j + 1]){
                temp = x[j];
                x[j] = x[j + 1];
                x[j + 1] = temp;
            }
            j = j + 1;
        }
        j = 0;
        i = i - 1;
    }
```

```
    i = 0;
    while(i < 5){
        write(x[i]);
        i = i + 1;
    }
    return 0;
}
```

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

```
int fact(int m){
    int result = 1;
    int j = 1;
    while(j <= m){
        result = result * j;
        j = j + 1;
    }
    return result;
}
int main(){
    int x[5];
    int i = 0, N = 5;
    while(i < N){
        x[i] = i + 2;
        write(fact(x[i]));
        i = i + 1;
    }
    return 0;
}
```

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

```
int hanoi(int n, int p1, int p2, int p3){
    if(n == 1){
        write(p1*1000000+p3);
    }
    else{
        hanoi(n-1,p1,p3,p2);
        write(p1*1000000+p3);
        hanoi(n-1,p2,p1,p3);
    }
    return 0;
}
int main(){
    int sum = 3;
    hanoi(sum, 1, 2, 3);
    return 0;
}
```

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

```
int gcd1(int a, int b)
{
    int result;
    int temp;
    temp = a - a / b * b;
    while (temp != 0) {
        a = b;
        b = temp;
        temp = a - a / b * b;
    }
    result = b;
    return result;
}
int gcd2(int x, int y)
{
    int remainder = 0;
    if(x > y){
        remainder = x - x / y * y;
        if(remainder == 0)
            return y;
        else
            return gcd2(y, remainder);
    }
    else {
        remainder = y - y / x * x;
        if(remainder == 0)
            return x;
        else
            return gcd2(x, remainder);
    }
}

```

```
int main()
{
    int m = read();
    int n = read();
    write(gcd1(m,n));
    write(gcd2(m,n));
    return 0;
}
```

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

```
int squaRever(int num)
{
    int flag = 0;
    int array[3];
    int j = 0;
    array[0] = num / 100;
    array[1] = num / 10 - 10 * array[0];
    array[2] = num - 100 * array[0] - 10 * array[1];
    if (array[0] != array[2]) {
        flag = 0;
    }
    else {
        while (j < 12){
            if ((j * j - num) == 0 )
                flag = 1;
            j = j + 1;
        }
    }
    if (flag == 1)
```

```
        return 1;
    else
        return 0;
}

int main()
{
    int i = 100;
    while (i < 150){
        if (squaRever(i) == 1)
            write(i);
        i = i + 1;
    }
    return 0;
}
```

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

```
int mod(int number2, int m2)
{
    int result = number2 - number2 / m2 * m2;
    int result2 = result;
    return result;
}

int power(int base1, int p1) {
```

```
    int ret1 = 1 + p1 - p1;
    while(p1 > (ret1 - ret1 + 90 - 89 + 1 - 2))
    {
        ret1 = ret1 * base1;
        p1 = 2 * 1 * p1 - 1 * p1 - 1;
    }
    return ret1;
}

int getNumDigits(int number3)
{
    int ret3 = 0;
    if(number3 < 0)
        return -1;
    while(number3 > 0) {

        number3 = number3 / 10;
        ret3 = ret3 + 2;
        ret3 = ret3 + 2;
        ret3 = ret3 - 3;
    }

    return ret3;
}

int isNarcissistic(int number4)
{
    int numDigits4 = getNumDigits(1 + number4 - 1);
    int sum4 = 0;
    int n4 = number4;
    int s4;
    while(n4>0) {
```

```
        s4 = mod(n4, 10);
        n4 = (n4 - s4) / 10;
        sum4 = sum4 + power(s4, numDigits4);
    }

    if(sum4 == number4)
        return 1;
    else
        return 0;
}

int main() {
    int count = 0;
    int i = 300;
    while(i < 500)
    {
        if(isNarcissistic(i) == 1)
        {
            write(i);
            count = count + 1;
        }
        i = i + 1;
    }
    write(count);
    return count;
}
```

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

```
int Josepy(int n,int m){
```

```
    int people[100], i = 1;
    int j = 0, k = 0, num = 0;
    while(i <= n){
        people[i] = 1;
        i = i + 1;
    }
    i = 1;
    while(i <= n){
        if(people[i] == 1){
            j = j + people[i];
            if(j == m){
                write(i);
                j = 0;
                people[i] = 0;
                k = k + 1;
            }
            if(k == n){
                num = i;
                return 0;
            }
        }
        if(i == n)
            i = 0;
        i = i + 1;
    }
    return 0;
}
int main(){
    int a, b;
    a = read();
    b = read();
    Josepy(a, b);
```

```
    return 0;
}
```

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。

```
int changeCal(int r1)
{
    int r2;
    r2 = r1 + 345 -345 + r1 * r2 - r1 * r2 + r1 * r1 - r1 * r1;
    r2 = r2 * 2 - r2 * 3 + r2 * 2 + 34 / 2 + 10 / 3 - 20 / 1;
    r2 = 7 + 6 - 11 * 1;
    r2 = r1 + 4 * 5 - 10 * 2 - 5 / 6;
    return r2;
}

int main()
{
    int a = 7 + 6 - 11 * 1, b = 8 - 4, c = 4 + 4 + 5 * 6 / 3 - 4;
    int d = a + b - c;
    int e = a + b + c * 2;
    int f = a + b + c;
    int g1 = 42, i = 0;
    int g, h;
    f = a + b + c - d - e + f;
    while (b - a < f) {
        g1 = g1 + i * 4 + 3 + 4 + 5;
        g = f - changeCal(b) + a * 2 + c * d - f;
        i = i + 1 + 0;
        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

```
struct Product{
    int weight;
    int price;
};

int main()
{
    struct Product p1;
    p1.weight = 123;
    p1.price = 56;
    write(p1.weight * p1.price);
    return 0;
}
```

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

```
struct Worker{
    int salary;
    int year;
    int esca;
};

int main(){
    struct Worker worker[10];
    int i, j, add, N = 10;
    int final = 0;
    while(i < N){
        worker[i].salary = 100;
```

```
        worker[i].year = i;
        worker[i].esca = i * i - i + 2;
        i = i + 1;
    }
    while(j < N){
        if(worker[j].year < 3)
            add = 1;
        else
            add = 2;
        final = final + worker[j].salary * (365 - worker[j].esca) *
           add;
        j = j + 1;
    }
    write(final);
    return 0;
}
```

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

```
struct Detail{
    int score;
    int name;
};

struct ScoreClass{
    struct Detail scoreDetail[100];
    int num;
};

```

```
int average(struct ScoreClass class){
    int scoreSum = 0;
    int i, N = class.num;
    while(i < N){
        scoreSum = class.scoreDetail[i].score + scoreSum;
        i = i + 1;
    }
    return scoreSum / N;
}

int main(){
    struct ScoreClass classInit;
    int result = 0, initN = 10;
    int j = 0;
    classInit.num = initN;
    while(j < initN){
        classInit.scoreDetail[j].score = (j + 1) * (j + 1);
        j = j + 1;
     }
    result = average(classInit);
    write(result);
    return 0;
}
```

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

```
int main(){
    int array[5][4][3];
    int r = 5, c = 4, d = 3;
```

```
    int i = 0, j = 0, k = 0, sum = 0;
    while(i < r){
        j = 0;
        while(j < c){
            k = 0;
            while(k < d){
                array[i][j][k] = (i + 1) * (j + 2) * (k + 3) + 4;
                k = k + 1;
            }
            j = j + 1;
        }
        i = i + 1;
    }
    i = 0;
    j = 0;
    k = 0;
    while(i < r){
        j = 0;
        while(j < c){
            k = 0;
            while(k < d){
                sum = array[i][j][k] + sum;
                k = k + 1;
            }
            j = j + 1;
        }
        i = i + 1;
    }
    write(sum);

    return 0;
}
```

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

```
int print(int array[3], int len){
    int i1 = 0;
    while(i1 < len){
        write(array[i1]);
        i1 = i1 + 1;
    }
    return 0;
}

int InnerP(int array1[3], int array2[3]){
    int result = 0;
    int i = 0, N = 3;
    while(i < N){
        result = result + array1[i] * array2[i];
        i = i + 1;
    }
    return result;
}

int main(){
    int p[3], q[3];
    int k = 0, M = 3;
    while(k < M){
        p[k] = k * k;
        q[k] = 2 * k + 1;
        k = k + 1;
```

```
    }
    print(p, 3);
    print(q, 3);
    write(InnerP(p,q));

    return 0;
}
```

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

```
int display(int chess2[16]){

    int chessplate[4][4];
    int i2 = 0, j2 = 0;
    while(i2 < 4){
        j2 = 0;
        while(j2 < 4){
            chessplate[i2][j2] = chess2[i2 * 4 + j2];
            j2 = j2 + 1;
        }
        i2 = i2 + 1;
    }
    i2 = 0;
    j2 = 0;
    while(i2 < 4){
        j2 = 0;
        while(j2 < 4){
            if(chessplate[i2][j2] == 1)
                write(j2);
```

```
            j2 = j2 + 1;
        }
        i2 = i2 + 1;
    }
    return 0;
}

int PutQueen(int chess[16], int a[4], int b[7], int c[7], int n, int
    sum)
{
    int col, i, j;
    col = 0;
    while(col < 4)
    {
        if(a[col] && b[n + col] && c[n - col + 3])
        {
            chess[n * 4 + col] = 1;
            a[col] = 0;
            b[n + col] = 0;
            c[n - col + 3] = 0;
            if(n == 3)
            {
                sum = sum + 1;
                display(chess);
            }
            else
                sum = PutQueen(chess, a, b, c, n+1, sum);

            chess[n * 4 + col]=0;
            b[n + col]=1;
            c[n - col + 3]=1;
            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;
```

```
83 }
```

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。