实验2

练习1

题目

求$\sqrt{100+\sqrt{99+\sqrt{98+\sqrt{\cdots+\sqrt{1}}}}}$的近似值。

解析

使用迭代法编写程序即可。

代码

```
public class Exercise1 {

    public static void main(String[] args) {
        double n = 0;
        for (int i = 1; i <= 100; i++) {
            n = Math.sqrt(n + i);
        }
        System.out.println(n);
    }

}
```

输入

```
无
```

输出

```
10.509990605994101
```

练习2

题目

根据$\frac{\pi}{2}=\frac{2}{\sqrt{2}}\times\frac{2}{\sqrt{2+\sqrt{2}}}\times\frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}}\times\cdots$求π的近似值，要求取前100个分式。

解析

同理，仅需在练习1的基础上稍加改动。

代码

```
public class Exercise2 {

    public static void main(String[] args) {
        double halfOfPi = 1;
        double n = 0;
        for (int i = 0; i < 100; i++) {
            n = Math.sqrt(2 + n);
            halfOfPi *= 2 / n;
        }
        double pi = 2 * halfOfPi;
        System.out.println(pi);
    }

}
```

输入

```
无
```

输出

```
3.141592653589797
```

练习3

题目

根据$\frac{\pi}{2}=1+\frac{1}{3}+\frac{1}{3}\times\frac{2}{5}+\frac{1}{3}\times\frac{2}{5}\times\frac{3}{7}+\frac{1}{3}\times\frac{2}{5}\times\frac{3}{7}\times\frac{4}{9}+\cdots$求π的近似值，要求取前100项。

解析

同理，仍然仅需在练习1的基础上稍加改动。

代码

```
public class Exercise3 {

    public static void main(String[] args) {
        double halfOfPi = 1;
        double n = 1;
        for (int i = 1; i < 100; i++) {
            n *= i / (double)(2 * i + 1);
            halfOfPi += n;
        }
        double pi = 2 * halfOfPi;
        System.out.println(pi);
    }

}
```

输入

```
无
```

输出

```
3.1415926535897922
```

练习4

题目

有两个两位正整数i和j，已知i减去j等于56，i^2的末两位数字等于j^2的末两位数字。求i和j的值。

解析

由于i、j都是两位正整数，而i-j=56，则j至少为10，i至少为66，j至多为43，i至多为99.

由于情况并不多，因此程序仅采用了暴力搜索的简单逻辑。

代码

```
public class Exercise4 {

    public static void main(String[] args) {
        for (int i = 66, j = 10; i <= 99; i++, j++) {
            if (i * i % 100 == j * j % 100) {
                System.out.println(String.format("i = %d, j = %d", i, j));
            }
        }
    }

}
```

输入

```
无
```

输出

```
i = 78, j = 22
```

练习5

题目

Counting Numbers

Starting from a positive integer n ($1 \le n \le 2001$). On the left of the integer n, you can place another integer m to form a new integer mn, where m must be less than or equal to half of the integer n. If there is an integer k less than or equal to half of m, you can place k on the left of mn to form a new integer kmn, …, and so on. For example, you can place 12 on the left of 30 to form an integer 1230, and you can place 6 to the left of 1230 to form an integer 61230, …, and so on. For example, start from n = 8, you can have the following 10 integers (including the integer you start with): 8, 18, 28, 38, 48, 128, 138, 148, 248, 1248.

Given an integer n, find the number of integers you can get using the procedure described above.

解析

简单的递归应用，无需更多说明。

代码

```
import java.util.Scanner;

public class Exercise5 {

    private static int count(int n) {
        if (n == 1)
            return 1;
        int result = 1;
        for (int i = 1; i <= n / 2; i++) {
            result += count(i);
        }
        return result;
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        in.close();
        System.out.println(count(n));
    }

}
```

输入

```
8
```

输出

```
10
```

心得体会

1. 当传统方法求解问题遇到变量过多问题是，可以使用迭代法。
2. 在许多情况下，使用递归能够简化逻辑，易于程序编写。