本章练习

第3章 控制语句与算法

引 言

Java语言与所有的程序设计语言一样，使用选择语句与循环语句控制程序的执行流程。本章主要介绍选择语句、循环语句的语法格式，以及这些语句的用法，并结合具体的例子介绍如何使用这些语句进行算法设计。

观看视频

3.1 Java程序的执行流程

3.1.1 算法的执行

Java语言是一种典型的面向对象程序设计语言，整个程序都是由对象组成的，对象中包含属性与方法两部分。在方法部分编程时，只有输入、计算与输出语句是无法实现复杂的算法功能的，例如，对于求和运算 $f(n)=\sum_{i=1}^{n}\frac{1}{i}$，求出 f(n)>=10 的最小 n。针对这个问题，可以设计如下的算法。

(1) 初始化变量："f=0,i=0;"。

(2) 如果 f<10，则转(3)，否则转(6)。

(3) i=i+1。

(4) f=f+1/i。

(5) 转(2)。

(6) n=i，返回 n。

从这个简单的算法可以看出，算法的执行是从上到下逐条顺序执行的，在这个顺序执行过程中，需要进行选择判断，并根据选择判断结果控制执行流程(语句)。语句(2)为判断语句，语句(3)～(5)是一个循环累加过程。

Java语言提供了实现上述基本控制结构的语句，有了这些流程控制语句，就可以为任何复杂算法编写程序。

3.1.2 语句块与块作用域

在Java程序中，语句块的概念很重要。语句块又称为复合语句，是指由一对{}括起来的0到多条语句。一个语句块中的语句可以看作一个整体，它们要么都执行，要么都不执

行，一个语句块可以嵌套在另一个语句块内。

语句块确定了块内变量的作用域。下面的代码是在 main()方法中嵌套一个语句块，在该语句块中定义了一个块内变量 i。

```
public static void main(String[] args) {
  float f = 0;
  …
  { int i = 0;
        …
  }//变量 i 只在本语句块内可见
        …
}
```

在上面的代码中，变量 f 在整个 main()方法内可见，即可以访问，而变量 i 只在嵌套的语句块内可见，在语句块外是不存在的。

Java 不允许在嵌套的两个块中声明同名的变量，例如，下面的代码会报编译错误。

```
public static void main(String[] args) {
      float f = 0;
      …
      { int i = 0;
      float f;                //编译出错，f 不能重复定义
        …
      }
        …
  }
```

3.2　选择语句

选择语句又称为条件语句、分支语句。该语句提供了基于条件的不同，选择不同执行路径的能力。Java 语言的选择语句有两种，分别是 if 语句与 switch 语句。

3.2.1　if 语句

if 语句有三种具体格式，分别是简单 if 语句、if-else 语句和复合 if 语句。图 3-1 给出了这三种 if 语句的流程图。

1. 简单 if 语句

简单 if 语句的格式为

```
if (<逻辑表达式>)
    <语句块>
```

其中，<逻辑表达式>的取值只能是 true 或 false，而且必须用圆括号括起来。该 if 语句的执行流程是先计算<逻辑表达式>的值，如果该值为 true，则执行<语句块>中的语句，否则不执行<语句块>。如果<语句块>只有一条语句，{}可以省略。例如：

```
if (a > b)
   System.out.println("a 大于 b");
```

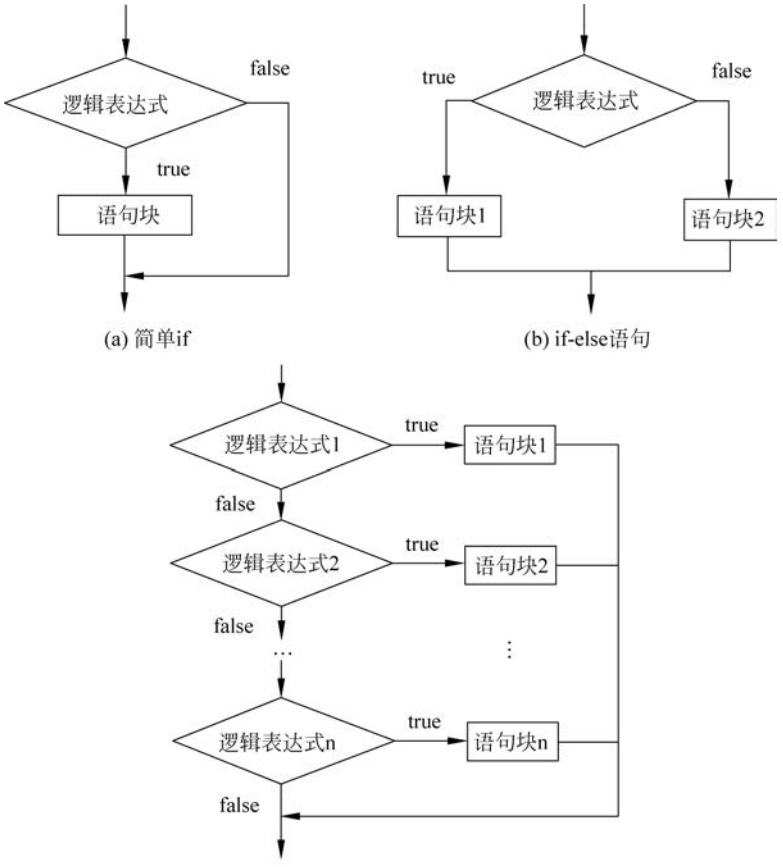

图 3-1　if 语句的流程图

与

```
if (a > b)
   {System.out.println("a 大于 b");}
```

是等价的。

2. if-else 语句

if-else 语句的格式为

```
if (<逻辑表达式>)
   <语句块 1 >
else
   <语句块 2 >
```

该语句执行时，首先计算<逻辑表达式>的值，如果该值为 true，则执行<语句块 1 >，否则执行<语句块 2 >。if-else 语句又称为二择一选择语句。这里<语句块 1 >与<语句块 2 >是指用{}括起来的 0 到多条语句，如果只有一条语句，{}可以省略。

3. 复合 if 语句

复合 if 语句又称为多择一 if 语句，其格式为

```
if (<逻辑表达式 1>)
    <语句块 1>
else if (<逻辑表达式 2>)
    <语句块 2>
  …
else
    <语句块 n>
```

该语句执行时，首先计算<逻辑表达式 1>的值，如果该值为 true，则执行<语句块 1>，否则计算<逻辑表达式 2>的值，如果该值为 true，则执行<语句块 2>，以此类推。如果所有的逻辑表达式的值都为假，则执行<语句块 n>。这种复合的 if 语句实际上是从多个条件中从上到下逐个判断，择一执行。

【例 3-1】 输入三个数，按由小到大的顺序输出。

```
import java.util.Scanner;
public class Example3_1 {
     public static void main(String [] args){
         float a,b,c,t;
         Scanner sc = new Scanner(System.in);
         a = sc.nextFloat();
         b = sc.nextFloat();
         c = sc.nextFloat();
         if (a > b)
               {t = a;
               a = b;
               b = t;
              }
         if (c < a)
               System.out.println("三个数由小到大:" + c + " " + a + " " + b );
         else if (c >= b)
               System.out.println("三个数由小到大:" + a + " " + b + " " + c );
         else
               System.out.println("三个数由小到大:" + a + " " + c + " " + b );
    }
}
```

在 Example3_1 中，用 Scanner 对象 sc 从键盘上读取三个数据，输入的三个数分别存到变量 a、b、c 中。在输入数据时，三个数据用空格分隔，最后按 Enter 键。下面是该程序运行时的数据输入与结果输出。

```
2  3  1
三个数由小到大: 1.0  2.0  3.0
```

程序中，if(a>b){…}是一个简单的 if 语句，该语句实现把两个数中大的放到 a，小的放到 b。第二个 if 语句是一个多择一的复合 if 语句。

3.2.2　switch 语句


观看视频

在处理多个选择时，使用多择一的复合 if 语句有时显得有些笨拙。为此，许多编程语言提供了 switch 语句，该语句的语法结构如下。

```
switch (< switch 表达式>) {
case value1:<语句块 1 >;
            break;
case value2:<语句块 2 >;
            break;
…
case valueN:<语句块 N >;
            break;
default:<其他语句块>;
}
```

图 3-2 给出了 switch 语句的执行流程。从该流程中可以看出，在 switch 语句执行时，首先计算< switch 表达式>的值 ，然后判断该值是否等于 value1，如果相等，则执行<语句块 1 >与 break；否则判断该值是否等于 value2，如果相等，则执行<语句块 2 >与 break，以此类推。如果该值与所有的 value 值都不等，则执行 default 后的<其他语句块>。

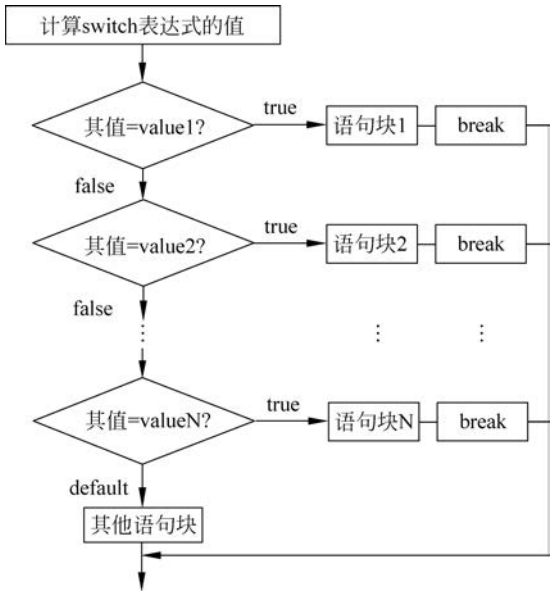

图 3-2　switch 语句的执行流程

这里需要特别说明的是在每个 case 语句末尾的 break 语句，break 语句的功能是跳出该 switch 语句，执行流程跳到 switch 之后的语句。如果在 case 之后没有 break 语句，程序会继续执行下一条 case 语句后的<语句块>，直到遇到 break 或到达 switch 的末尾。

switch 语句有如下规则。

（1）switch 语句可以拥有多条 case 语句。每个 case 后面跟一个要比较的值和冒号。

（2）switch 语句中的 switch 表达式的类型只能为 byte、short、int、char 与字符串，且 case 语句中值的数据类型必须与之相同。

（3）当 switch 表达式的值与 case 语句的值相等时，case 语句之后的语句开始执行，当遇到 break 语句时，程序跳转到 switch 语句后面的语句执行。

（4）case 语句不必须包含 break 语句，但如果没有 break 语句，程序会继续执行下一条 case 语句后的语句块。

(5) switch 语句的最后可以包含一个 default 分支,其含义是在没有 case 语句的值和 switch 表达式值相等时,执行其后的<其他语句块>。

【例 3-2】 编写一个程序,把百分制的成绩转换成五级分制。

在学生成绩登记中,百分制向五级分制的转换规则是 90～100 分为优、80～89 分为良、70～79 分为中、60～69 分为及格、60 分以下为不及格。下面的程序就是按照这个规则实现转换的。

```
import java.util.Scanner;
public class Example3_2 {
     public static String ptoword(float grad) {              /* 输入分数转换为文字 */
     String result = null;
     switch ((int)grad/10) {
     case 10:
     case 9:
     result = "优";
          break;
     case 8:
          result = "良";
          break;
          case 7:
          result = "中";
          break;
     case 6:
          result = "及格";
          break;
          default:
          result = "不及格";
          }
          return result;
    }
public static void main(String[] args)
  {float point = 0;
    Scanner scan = new Scanner(System.in);
    while (true)
        {
            System.out.println("Please input score 0 -- 100:");
            point = scan.nextFloat();
            if (point >= 0 & point <= 100)
            System.out.println("The point is: " + ptoword(point));
            else
            System.out.println("The score should be 0 -- 100");
     }
  }
}
```

3.3　循环语句


观看视频

当程序中出现了大量的重复且有规律的代码段时,可以用循环语句控制重复代码段的执行。例如,如果需要输出 100 次“欢迎,欢迎,热烈欢迎”,不用循环语句会写成下面的

形式。

```
System.out.println("1. 欢迎,欢迎,热烈欢迎!");
System.out.println("2. 欢迎,欢迎,热烈欢迎!");
    …
System.out.println("100. 欢迎,欢迎,热烈欢迎!");
```

显然这种写法显得太笨拙了,如果使用循环语句,代码会变得非常简洁。上面的代码用循环语句实现如下：

```
for (int i = 1;i <= 100;i++)
    System.out.println(i + ". 欢迎,欢迎,热烈欢迎!");
```

Java 语言的循环语句有三种,分别是 for 循环、while 循环与 do-while 循环。三种循环语句各有所适应的循环场景,编程人员可以根据实际问题选择合适的循环语句。

3.3.1 while 语句

while 语句是一种常用的循环语句,其语法格式如下。

```
while (<逻辑表达式>) {
    <循环体>;
}
```

其中,<逻辑表达式>是循环控制条件,其值是布尔值 true 或 false。在循环开始执行时,先计算一次<逻辑表达式>的值,若条件为 false,则 while 语句结束；否则执行循环体中的语句并返回到 while 的开始,继续计算<逻辑表达式>并根据该值确定是否继续循环。图 3-3 给出了 while 语句的执行流程。

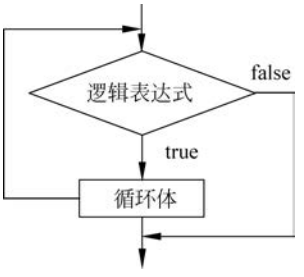

图 3-3 while 语句的执行流程

【例 3-3】 $f(n)=\sum_{i=1}^{n}\frac{1}{i}$,编程求出 f(n)≥10 的最小 n。

```
public class SmallestN {
    public static void main(String[] args) {
        double f = 0;
        long n,i = 0;
        while(f < 10){
            i = i + 1;
            f = f + 1.0/i;
        }
        n = i;
        System.out.println("Smallest n = " + n);
        System.out.println("f(n) = " + f);
    }
}
```

该程序的运行结果如下。

```
Smallesst n = 12367
f(n) = 10.000043008275778
```

3.3.2　do-while 语句

do-while 语句的语法格式如下：

```
do
        <循环体>
while(<逻辑表达式>);
```

其中，<逻辑表达式>是循环控制条件，其值是布尔类型。do-while 语句的执行与 while 语句稍有不同。在 while 语句中，如果一开始的<逻辑表达式>为 false，则循环体一次都不执行；而 do-while 语句的执行流程是先执行循环体，然后再计算<逻辑表达式>，如果该值为 true，则继续循环，直到该值为 false。因而，do-while 语句至少执行一次<循环体>中的语句。这里的<循环体>通常是一个语句块，do-while 语句的执行流程见图 3-4。

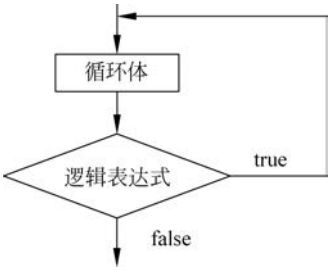

图 3-4　do-while 语句的执行流程

【例 3-4】 从键盘上输入若干数，求这组数据的平方和 s，当平方和 s＞10^8 或者输入的数为 0 时，输出数据个数与平方和 s，程序结束。

```
import java.util.Scanner;
public class Example3_4 {
     public static void main(String[] args) {
          double s = 0,x;
          int i = 0;
          Scanner sc = new Scanner(System.in);
          do {
              x = sc.nextDouble();
              i = i + 1;
              s = s + Math.pow(x,2);
          } while (x!= 0 && s <= 1.0e + 8);
          System.out.println("输入数据的平方和 = " + s);
          System.out.println("输入数据个数 = " + i);
    }
}
```

在以上程序中，用 Scanner 对象 sc 从键盘上读取 double 类型的值并赋值给 x，然后计算 x 的平方并累加到 s 中，接下来计算逻辑表达式(x!＝0 && s<＝1.0e＋8)的值，如果该值为 true 则继续循环，否则，循环结束，执行后面的输出语句。

需要注意的是，判断输入的 double 类型 x 是否为 0，应当用 Math.abs(x)小于一个足够小的数(这里用 1.0e－10，即 10^{-10})，最好不用 x!＝0，这是因为在编程语言中，由有限的二进制位数表示浮点数会存在误差，为了代码安全，当比较两个浮点数是否相等时，通常不用“＝＝”判断，而是用它们的差足够小来判断。

Math.pow(x,2)、Math.abs(x)是 Math 类中的静态方法，分别是计算 x^2 与|x|。

3.3.3 for 语句

for 语句是一种常用的循环语句，语法形式如下：

```
for(<初始化表达式>; <循环条件表达式>; <循环后操作表达式>)
        <循环体>
```


观看视频

for 循环把循环的初始化操作、循环条件、每次循环后的操作都放在 for 后面的圆括号中，表达式之间用英文分号分隔。这里的＜循环体＞为迭代执行的语句块，是每次循环执行的语句序列。for 语句的执行流程如图 3-5 所示，具体的执行过程如下。

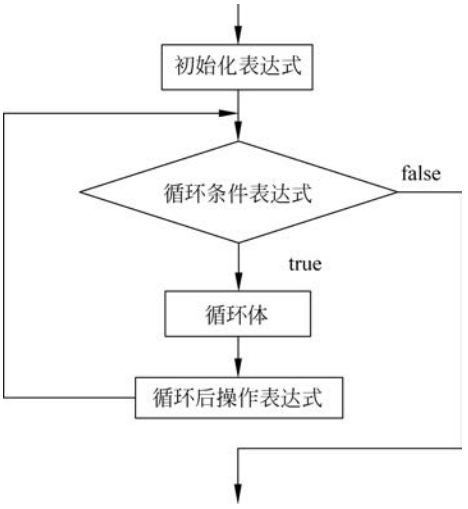

图 3-5　for 语句能够执行流程

第一步，先执行初始化表达式操作，对循环变量赋初值。在整个循环过程中，对循环变量赋初值操作只执行一次。

第二步，计算<循环条件表达式>的值。若该值为 true，表示循环条件成立，则执行<语句块>，然后执行<循环后操作表达式>，再回到本步的开头进行下一次循环。若该值为 false，则循环执行结束，执行流程跳到 for 循环的后继语句。

for 语句中的三个表达式可以都为空，即 for(;;)，表示是一个无限循环（又称死循环）。在这种情况下，循环体中需要根据执行情况，用 break 语句控制跳出循环，否则，程序会进入死循环。

【例 3-5】　编程计算 $1+\frac{1}{1!}+\frac{1}{2!}+\cdots+\frac{1}{10!}$。

```
public class Example3_5 {
    public static void main(String[] args) {
        double s = 1;
        long f = 1;
        for (int i = 1;i <= 10;i++)
          {f = f * i;
            s = s + 1.0/f;
          }
```

```
            System.out.println("Eular number = " + s);
        }
    }
```

在程序中，用 f 存放 i! 的值，即当 i=1 时，f 中是 1!；在下次循环时，当 i 变成 2 时，执行 f=f * i，把 2! 赋值给 f，以此类推，当 i 循环到 10 时，f 中存放的是 10!。

3.4　跳转语句

在循环语句中使用 break 与 continue 语句可以提供对循环过程的额外控制。在某些情况下，使用 break 与 continue 语句可以简化编程。尽管 Java 提供了这种跳转语句，但由于使用它们可能会使程序难以阅读和调试，在编程时，尽量不要使用它们。

3.4.1　break 语句

在循环执行过程中，如果想在特定条件下退出循环的执行，可以调用 break 语句。下面的程序给出了在循环中使用 break 语句的效果。

```
public class BreakDemo {
    public static void main(String[ ] args) {
        int sum = 0;
        int number = 0;
        while (number < 20) {
            number++;
            sum += number;
            if (sum >= 100)
                break;
        }
            System.out.println("The number is " + number);
            System.out.println("The sum is " + sum);
        }
    }
```

将整数从 1 到 20 相加，直到和大于或等于 100 时，循环结束。如果没有 if 语句，程序将计算 1～20 的数字之和。在使用 if 语句时，当和 sum 大于或等于 100 时，break 语句控制执行流程跳出循环。

3.4.2　continue 语句

continue 语句可以用在任何循环控制结构中，其功能是跳过循环体中 continue 后剩余的语句而强行执行下一次循环，即终止当前迭代，进入下一次循环。它用在 for 循环体与 while 循环体中，执行过程略有不同。

(1) 在 for 循环中，continue 语句将导致控制流程立即跳到<循环后操作表达式>，更新循环变量，然后进行下一次循环条件判断。

(2) 在 while 循环或者 do-while 循环中，continue 语句控制流程立即跳到<逻辑表达式>计算，并根据该值决定是否继续循环。

下面的程序对1～99中的每个数进行判断，如果这个数是7的倍数或者个位为7，则执行continue语句，跳转到while <逻辑表达式>计算，并根据该<逻辑表达式>的值决定是否继续循环。

```
public class ContinueDemo {
public static void main(String[] args) {
    int sum = 0;
    int number = 0;
    while (number < 100) {
        number++;
        if (number % 7 == 0 || number % 10 == 7)
            continue;
        sum += number;
    }
    System.out.println("The sum is " + sum);
}
}
```

该程序最终的运行输出是“The sum is 3879”，该值是1～99中不是7的倍数并且末位不是7的所有数字之和。

3.5 控制语句编程举例

本节通过例子介绍常用的算法设计。

【例3-6】 计算一元二次方程 $ax^2+bx+c=0$ 的根，系数a、b、c从键盘输入。

该程序需要考虑以下几种情况。

(1) a、b都为0，程序应输出“不是方程”信息。

(2) a为0并且b不为0，程序是一元一次方程。

(3) a、b都不为0，这时要判断 b^2-4ac 的值是否大于或等于0。若 $b^2-4ac\geqslant 0$，则输出两个实数，否则，显示方程“没有实数根”。

程序编写如下：

```
import java.util.*;
public class Example3_6 {
    public static void main(String[] args) {
        double a,b,c,x1,x2;
        Scanner scan = new Scanner(System.in);
        System.out.println("input a,b,c: ");
        a = scan.nextDouble();
        b = scan.nextDouble();
        c = scan.nextDouble();
        if (a == 0)
            if (b == 0)
                System.out.println(" a,b 都为 0,不是方程");
            else
                System.out.println(" a 为 0,只有一个根 x = " + (-c/b));
        else
           if (b * b - 4 * a * c >= 0)
```

```
                { x1 = ( - b + Math.sqrt(b * b - 4 * a * c))/2/a;
                  x2 = ( - b - Math.sqrt(b * b - 4 * a * c))/2/a;
                  System.out.println("x1 = " + x1 + " x2 = " + x2);
                }
            else
                  System.out.println("没有实数根");
    }
}
```

在该程序中，在 if(a==0)…else…之后各嵌入了一条 if-else 语句。在这种 if-else 的嵌套中，if-else 的配对方法是 else 总是和上方还没有配对的 if 配对。

【例 3-7】 求两个正整数的最大公约数。

对于从键盘上输入的两个整数 n1 与 n2，需要先判断是不是都是正整数，如果不是，则显示"n1 和 n2 必须是大于 0 的整数"。如果两个数都是正整数，程序中先把两个数的较小者赋值给变量 n。由于两个数的最大公约不会大于变量 n 的值，因而可以构建循环，判断变量 n，变量 n－1，…，1 能不能整除 n1 和 n2。编写程序如下：

```
import java.util.Scanner;
public class Example3_7 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("输入第一个整数: ");
        int n1 = input.nextInt();
        System.out.print("输入第二个整数: ");
        int n2 = input.nextInt();
        int gcd = 1;
        int n;
        if (n1 > 0 && n2 > 0) {
            if (n1 >= n2)
                n = n2;
            else
                n = n1;
            while (n>=1) {
                if (n1 % n == 0 && n2 % n == 0)
                {gcd = n;
                 break;
                }
                n--;
            }
            System.out.println( n1 + " 和 " + n2 + " 最大公约数是: " + gcd);
        } else
            System.out.println("n1 和 n2 必须是大于 0 的整数");
    }
}
```

【例 3-8】 在屏幕上输出九九乘法口诀表。

要输出一个九九乘法口诀表，需要二重循环。程序代码与输出如下：

```
import java.util.Scanner;
public class Example3_8 {
```

```
public static void main(String[] args) {
    int i,j,n;
    Scanner scan = new Scanner(System.in);
    System.out.print("输入 n");
    n = scan.nextInt();
    System.out.println("                    九九乘法口诀表");
    i = 1;
    System.out.print(" ");
    while (i <= n)
    {System.out.printf("%6d",i);
        i++;
    }
    System.out.println();
    i = 1;
    while (i <= n)
    {System.out.print("--------");
        i++;
    }
    System.out.println();
    i = 1;
    while (i <= n)
    {System.out.printf("%2d",i);
        System.out.print("| ");
        j = 1;
        while (j <= n)
        {
            System.out.printf("%6d",i * j);              // 使用格式化输出
            j++;
        }
        System.out.println();
        i++;
    }
}
}
```

为了使输出的九九乘法口诀表每列对得整齐，需要让每列数据占相同的宽度，为此程序中使用 printf("%6d",x)格式化输出整数 x，其含义是以 6 为宽度、右对齐的格式输出整数 x。程序的运行结果如下：

```
输入 n: 10
                                九九乘法口诀表
        1      2      3      4      5      6      7      8      9     10
-------------------------------------------------------------------------
  1|    1      2      3      4      5      6      7      8      9     10
  2|    2      4      6      8     10     12     14     16     18     20
  3|    3      6      9     12     15     18     21     24     27     30
  4|    4      8     12     16     20     24     28     32     36     40
  5|    5     10     15     20     25     30     35     40     45     50
  6|    6     12     18     24     30     36     42     48     54     60
  7|    7     14     21     28     35     42     49     56     63     70
  8|    8     16     24     32     40     48     56     64     72     80
  9|    9     18     27     36     45     54     63     72     81     90
 10|   10     20     30     40     50     70     70     80     90    100
```

【例 3-9】 求所有的水仙花数。

水仙花数是指一个 3 位整数，它的每个位上的数字的 3 次幂之和等于它本身，例如：$1^3+5^3+3^3=153$。本程序采用穷举法，逐个测试三位整数(100～999)是不是水仙花数。程序代码如下：

```
public class Example3_9 {
    public static void main(String[] args) {
        System.out.print("水仙花数有:");
        int i,j,k;
        int num1,num2;
        for (i = 1;i <= 9;i++)
          for (j = 0;j <= 9;j++)
           for (k = 0;k <= 9;k++){
              num1 = (int)(Math.pow(i,3) + Math.pow(j,3) + Math.pow(k,3));
              num2 = i * 100 + j * 10 + k;
              if (num1 == num2)
                System.out.printf(" % 7d",num2);
             }
        }
    }
```

运行该程序，输出结果如下：

```
水仙花数有: 153   370   371   407
```

【例 3-10】 输出斐波那契数列的前 20 项。

斐波那契数列指的是这样一个数列：1,1,2,3,5,8,13,21,34,55,89,…，这个数列从第 3 项开始，每一项都等于前两项之和。基于这一描述，可以用循环迭代的方法，每次循环中，用序列中前面两个值计算当前值，输出该序列，程序代码如下：

```
public class Example3_10 {
    public static void main(String[] args) {
       int a = 1;
       int b = 1;
       int t;
       System.out.printf(" % 6d % 6d",a,b);
       for (int i = 2;i <= 20;i++){
           t = a + b;
           System.out.printf(" % 6d",t);
           a = b;
           b = t;
        }
    }
}
```

对于斐波那契数列，还可以采用如下的递归形式进行定义。

$$f(n)=\begin{cases}1, & n=1 \text{ 或 } n=2\\ f(n-1)+f(n-2), & n>2\end{cases}$$

Java 允许递归调用，即一个方法直接或间接地调用自己。采用递归编程，程序代码如下。

```
public class FibonacciRecursion {
    public static int fib(int n){
        if (n<=0)
            return -1;
        else if (n<=2)
            return 1;
        else
            return fib(n-1)+fib(n-2);
    }
    public static void main(String[] args){
        int i;
        System.out.println("斐波那契数列:");
        for (i=1;i<=20;i++)
            System.out.printf("%6d",fib(i));

    }
}
```

在编程时，虽然递归算法使程序看来非常简洁，但是递归程序需要频繁地进行方法调用，运行效率比较低，计算时间比较长。

习题

1. 阅读以下 Java 语言程序的片段，写出程序的输出结果。

```
int t = 198;
do{
   System.out.println(t);
   t = t / 2;
} while(t > 0);
System.out.println("t = " + t);
```

2. 阅读以下 Java 语言程序的片段，写出程序的输出结果。

```
for (int i = 1; i < 4; i++) {
  for (int j = 1; j < 4; j++) {
      if (i * j > 2)
      continue;
      System.out.println(i * j);
      }
  System.out.println(i);
}
```

3. 把下面的代码转换成 switch 语句。

```
if (a == 1)
    x += 5;
else if (a == 2)
    x += 10;
else if (a == 3)
    x += 16;
```

```
else if (a == 4)
    x += 34;
else
     x += 100;
```

4. while 循环和 do-while 循环的区别是什么？将以下循环转换为 do-while 循环执行。

```
int sum = 0;
int number = input.nextInt();
while (number != 0) {
sum += number;
number = input.nextInt();
}
```

5. 在循环中，break 与 continue 的作用是什么？下面的两程序段哪个存在问题？为什么？

```
int x = 200;
while(true){
if(x<9)
break;
x = x - 8;
}
System.out.println("x is" + x);
```

```
int x = 200;
while(true){
if(x<0)
continue;
x = x - 8;
}
System.out.println("x is" + x);
```

6. Math.random()可以产生 0～1 随机的小数，通过运算可以把该数转换成 0～100 的随机整数。先编程产生 0～100 的随机整数 x，然后让用户输入对该数的猜测值 guess。如果 guess 等于 x，则显示“猜测正确”，否则显示 guess 值是大了还是小了，提示用户继续猜测，直到猜测正确为止。

7. 编写一个程序，从键盘读取一个整数，编写程序显示其所有最小的因子。例如，如果输入整数为 48，则输出如下：2、2、2、2、3。

8. 从键盘输入一个整数，判断并显示该数是不是素数。

9. 回文素数是指一个整数 n 从左向右和从右向左读其结果值相同且是素数。编程求出 1000～100 000 的回文素数。

10. 输入三角形的三条边 a、b、c，判断并输出能否构成三角形（任意两条边的和大于第三条边），如果能构成三角形，输出是什么类型的三角形（等边、等腰、直角）。

11. 编写一个程序，提示用户输入一个 1～20 的数，并显示一个金字塔，示例运行结果如下所示。

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe" …
Please input n:6
                                    1
                                  2 1 2
                                3 2 1 2 3
                              4 3 2 1 2 3 4
                            5 4 3 2 1 2 3 4 5
                          6 5 4 3 2 1 2 3 4 5 6
```

12. 编写一个程序，提示用户输入'A'～'Z'中的一个字符，并显示一个金字塔。例如，输入 H，则显示由 A 到 H 形成的金字塔，示例如下。

```
"C:\Program Files\Java\jdk1.8.0_281\bin\java.exe"…
请输入字符：H
                    A
                  B A B
                C B A B C
              D C B A B C D
            E D C B A B C D E
          F E D C B A B C D E F
        G F E D C B A B C D E F G
      H G F E D C B A B C D E F G H
```

13. 给出递归计算 n! 的方法 factorial(int n)，并编写主程序调用该递归方法。

14. 用 Java 语言编写程序，计算 e=1+1/1!+1/2!+…+1/n!。要求 e 值精确到小数点后第 6 位。试比较 n! 计算调用递归方法与不用递归方法的运行时间差异。

15. 完全数又称完美数，即如果一个正整数等于它的所有正因数的和，则它被称为完全数。例如，6 是第一个完全数，因为 6=3+2+1，下一个完全数是 28(28=14+7+4+2+1)。编写一个程序来找到 1～10 000 所有的完全数。

16. 在数学上可以用下面的公式计算 π。

$$\pi=4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots+\frac{1}{2i-1}-\frac{1}{2i+1}\right)$$

编程显示 i=100、1000、10 000、100 000 时的 π 值。

17. 编程统计全班的 90～100 分、80～89 分、70～79 分、60～69 分、不及格的人数。学生成绩由键盘输入，输入－1 表示结束。