第 3 章　程序流程控制

流程控制是所有编程语言的基本功能，主要用来控制程序中各语句的执行顺序。在 Java 语言中最主要的流程控制方式是结构化程序设计中规定的三种基本控制结构，即顺序结构、分支结构和循环结构。

顺序结构比较简单，它的执行过程是从所描述的第一个操作开始，按顺序依次执行后续的操作，直到序列的最后一个操作。前面的例子都是顺序结构的。本章主要讨论分支结构和循环结构。

3.1　分 支 结 构

分支结构也叫选择结构。Java 语言提供了两种分支结构，即 if-else 结构和 switch 结构。

3.1.1　if 语句结构

1. 单分支结构

if-else 结构是最常用的分支结构，可以实现单分支和双分支结构。单分支的 if 结构的一般格式如下：

```
if (condition){
    statements;
}
```

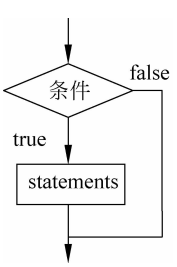

图 3-1　单分支结构

其中 condition 为布尔表达式，它的值为 true 或 false。程序执行的流程是：首先计算 condition 表达式的值，若其值为 true，则执行 statements 语句序列，否则转去执行 if 结构后面的语句，如图 3-1 所示。

编写程序，从键盘上读取一个整数，检查该数是否能同时被 5 和 6 整除，是否能被 5 或被 6 整除，是否只能被 5 或只能被 6 整除。

程序 3.1　CheckNumber.java

```
import java.util.Scanner;
public class CheckNumber{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    System.out.print("请输入一个整数: ");
    intnum = sc.nextInt() ;
```

```
    if(num % 5 == 0 && num % 6 == 0){
      System.out.println( num + " 能被 5 和 6 同时整除.") ;
    }

    if(num % 5 == 0 || num % 6 == 0){
      System.out.println( num + " 能被 5 或 6 整除.") ;
    }

    if(num % 5 == 0 ^ num % 6 == 0){
      System.out.println( num + " 能只被 5 或只被 6 整除.") ;
    }
```

下面是程序运行的一次结果：

```
请输入一个整数: 12
12 能被 5 或 6 整除.
12 能只被 5 或只被 6 整除.
```

2. 双分支结构

双分支的 if 结构的一般格式如下：

```
if (condition){
    statements1;
}else{
    statements2;
}
```

该结构的执行流程是，首先计算 condition 的值，如果为 true，则执行 statements1 语句序列，否则执行 statements2 语句序列，如图 3-2 所示。当 if 或 else 部分只有一条语句时，大括号可以省略，但推荐使用大括号。

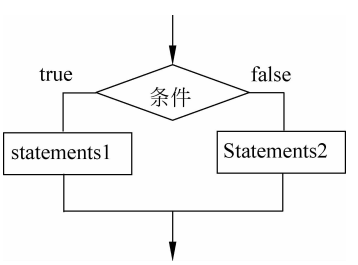

图 3-2　双分支结构

下面的程序要求从键盘上输入一个年份，输出该年是否是闰年。符合下面两个条件之一的年份即为闰年：

① 能被 4 整除，但不能被 100 整除。

② 能被 400 整除。

程序 3.2　LeapYear.java

```
import java.util.Scanner;
public class LeapYear{
  public static void main(String[ ] args){
    Scanner sc = new Scanner(System.in);
    System.out.print("请输入年份: ");
    int year = sc.nextInt() ;
    if((year % 4 == 0 && year % 100 != 0)||year % 400 == 0){
      System.out.println(year + " 年是闰年.") ;
    }else{
        System.out.println(year + " 年不是闰年.") ;
    }
  }
```

```
}
```

程序输出结果为:

```
请输入年份: 2013
2013 年不是闰年.
```

3. 阶梯式 if-else 结构

如果程序逻辑需要多个选择,可以在 if 语句中使用一系列的 else 语句,这种结构有时称为阶梯式 if-else 结构。下面程序要求输入学生的百分制成绩,打印输出等级的成绩。等级规定为,90 分(包括)以上的为“优秀”,80 分(包括)以上的为“良好”,70 分(包括)以上的为“中等”,60 分(包括)以上的为“及格”,60 分以下为“不及格”。

程序 3.3　ScoreGrade.java

```
import java.util.Scanner;

public class ScoreGrade{
  public static void main(String[] args){
    Scanner sc = new Scanner(System.in);
    System.out.print("请输入成绩: ");
    double score = sc.nextDouble() ;
    String grade = "";
    if(score > 100 || score < 0){
       System.out.println("输入的成绩不正确.");
       System.exit(0);       //结束程序运行
    }else if(score >= 90){
       grade = "优秀";
    }else if(score >= 80){
       grade = "良好";
    }else if(score >= 70){
       grade = "中等";
    }else if(score >= 60){
       grade = "及格";
    }else{
       grade = "不及格";
    }
    System.out.println("你的成绩为: " + grade);
  }
}
```

下面是程序的一次运行结果:

```
请输入成绩: 78
你的成绩为: 中等
```

3.1.2　条件运算符

条件运算符(conditional operator)的格式如下:

```
condition ? expr1: expr2
```

因为有三个操作数，又称为三元运算符。这里 condition 为关系或逻辑表达式，其计算结果为布尔值。如果该值为 true，则计算表达式 expr1 的值，并将计算结果作为整个条件表达式的结果；如果该值为 false，则计算表达式 expr2 的值，并将计算结果作为整个条件表达式的结果。

条件运算符可以实现 if-else 结构。例如，若 max，a，b 是 int 型变量，下面结构：

```
if (a > b) {
  max = a;
}
else {
  max = b;
}
```

用条件运算符表示为：

```
max = (a > b)? a : b ;
```

从上面可以看到使用条件运算符会使代码简洁，但是不容易理解。现代的编程，程序的可读性变得越来越重要，因此推荐使用 if-else 结构，毕竟并没有多输入多少代码。

3.1.3 switch 语句结构

如果需要从多个选项选择其中一个，可以使用 switch 语句。switch 语句主要实现多分支结构，一般格式如下：

```
switch (expression){
  case value1:
    statements    [break;]
  case value2:
    statements    [break;]
  ⋮
  case valuen:
    statements    [break;]
  [default:
   statements]
}
```

其中 expression 是一个表达式，它的值必须是 byte、short、int、char、enum 类型或 String 类型。case 子句用来设定每一种情况，后面的值必须与表达式值类型相容。程序进入 switch 结构，首先计算 expression 的值，然后用该值依次与每个 case 中的常量(或常量表达式)的值进行比较，如果等于某个值，则执行该 case 子句中后面的语句，直到遇到 break 语句为止。

break 语句的功能是退出 switch 结构。如果在某个情况处理结束后就离开 switch 结构，则必须在该 case 结构的后面加上 break 语句。

default 子句是可选的，当表达式的值与每个 case 子句中的值都不匹配时，就执行 default 后的语句。如果表达式的值与每个 case 子句中的值都不匹配，且又没有 default 子句，则程序不执行任何操作，而是直接跳出 switch 结构，执行后面的语句。

编写程序,从键盘输入一个年份(如 2000 年)和一个月份(如 2 月),输出该月的天数(为 29)。

程序 3.4　SwitchDemo.java

```
import java.util.Scanner;

public class SwitchDemo{
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Enter a year: ");
    int year = input.nextInt();
    System.out.print("Enter a month: ");
    int month = input.nextInt();
    int numDays = 0;
    switch (month) {
       case 1: case 3: case 5:
       case 7: case 8: case 10:
       case 12:
            numDays = 31;
            break;
       case 4: case 6: case 9: case 11:
            numDays = 30;
            break;
       case 2:     // 对于 2 月需要判断是否是闰年
            if (((year % 4 == 0) &&
                !(year % 100 == 0))
                || (year % 400 == 0))
               numDays = 29;
            else
               numDays = 28;
            break;
        default:
             System.out.println("月份非法.");
             break;
        }
        System.out.println("该月的天数为: "   + numDays);
    }
}
```

下面是程序的一次运行结果:

```
Enter a year: 2000
Enter a month: 2
该月的天数为: 29
```

在 Java SE 7 中,可以在 switch 语句的表达式中使用 String 对象,下面代码根据月份的字符串名称输出数字月份。

程序 3.5　StringSwitchDemo.java

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

```
    public static void main(String[ ] args) {
        String month = "";
        int monthNumber = 0;
        Scanner input = new Scanner(System.in);
        System.out.println("请输入一个月份的英文名称: ");
        month = input.next();
        switch (month.toLowerCase()) {
           case "january": monthNumber = 1;break;
           case "february": monthNumber = 2;break;
           case "march": monthNumber = 3; break;
           case "april": monthNumber = 4; break;
           case "may": monthNumber = 5; break;
           case "june": monthNumber = 6; break;
           case "july": monthNumber = 7; break;
           case "august": monthNumber = 8; break;
           case "september": monthNumber = 9; break;
           case "october": monthNumber = 10; break;
           case "november": monthNumber = 11; break;
           case "december": monthNumber = 12; break;
           default:
              monthNumber = 0; break;
        }

        if (monthNumber == 0) {
          System.out.println("输入的月份名非法");
        }else {
          System.out.println(month + "是" + monthNumber + "月");
        }
    }
}
```

程序中 month.toLowerCase()是将字符串转换成小写字符串。switch 表达式中的字符串与每个 case 中的字符串进行比较。

3.2 循环结构

在程序设计中,有时需要反复执行一段相同的代码,这时就需要使用循环结构来实现。Java 语言提供了 4 种循环结构:while 循环、do-while 循环、for 循环和增强的 for 循环。

一般情况下,一个循环结构包含 4 部分内容。

① 初始化部分:设置循环开始时的程序状态。

② 循环条件:循环条件一般是一个布尔表达式,当表达式的值为 true 时执行循环体,为 false 时退出循环。

③ 迭代部分:改变变量的状态。

④ 循环体部分:需要重复执行的代码。

3.2.1 while 循环结构

while 循环是 Java 最基本的循环结构,这种循环是在某个条件为 true 时,重复执行一个语句或语句块。它的一般格式如下:

```
[initialization]
while (boolean_expression ){
    // 循环体
    [iteration]
}
```

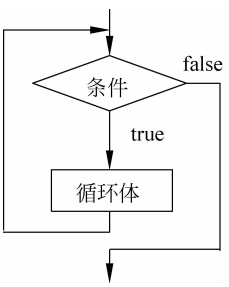

图 3-3　while 循环结构

其中,initialization 为初始化部分;boolean_expression 为一个布尔表达式,是循环条件;中间的部分为循环体,用一对大括号定界;iteration 为迭代部分。

该循环首先判断循环条件,当条件为 true 时,一直反复执行循环体。这种循环一般称为“当循环”。一般用在循环次数不确定的情况下。while 循环的执行流程如图 3-3 所示。

下面一段代码使用 while 结构求 1～100 之和。

```
int n = 1;
int sum = 0;
while(n <= 100){
  sum = sum + n;
  n = n + 1;
}
System.out.println("sum = " + sum);     // 输出 sum = 5050
```

程序 3.6 采用 while 循环结构,计算下面级数之和:

$$\frac{1}{3}+\frac{3}{5}+\frac{5}{7}+\frac{7}{9}+\frac{9}{11}+\frac{11}{13}+\cdots+\frac{95}{97}+\frac{97}{99}$$

程序 3.6　SeriesSum.java

```
public class SeriesSum{
  public static void main(String[] args){
    int n = 1;
    double sum = 0;
    while( n <= 99){
      sum = sum + (double) n / (n + 2);              // 将一个操作数转换成 double
      n = n + 2;
    }
    System.out.println("sum = " + sum);
  }
}
```

程序输出结果为:

```
sum = 46.10464832285218
```

下面的程序随机产生一个 100～200 的整数,用户从键盘上输入所猜的数,程序显示是否猜中的消息,如果没有猜中要求用户继续猜,直到猜中为止。

程序 3.7　GuessNumber.java

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

```
  public static void main(String[] args){
    int magic = (int)(Math.random() * 101) + 100;
    Scanner sc = new Scanner(System.in);
    System.out.print("请输入你猜的数: ");
    int guess = sc.nextInt();
    while(guess != magic){
      if(guess > magic)
        System.out.print("错误!太大,请重猜: ");
      else
        System.out.print("错误!太小,请重猜: ");
      guess = sc.nextInt();
    }
    System.out.println("恭喜你,答对了!\\n 该数是: " + magic);
  }
}
```

程序中使用了 java.lang.Math 类的 random 方法,该方法返回一个 0.0～1.0(不包括 1.0)的 double 型的随机数。程序中该方法乘以 101 再转换为整数,得到 0～100 的整数,再加上 100,则 magic 的范围就为 100～200 的整数。

3.2.2 do-while 循环结构

do-while 循环的一般格式如下:

```
[initialization]
do{
  // 循环体
  [iteration]
}while(termination);
```

do-while 循环执行过程如图 3-4 所示。

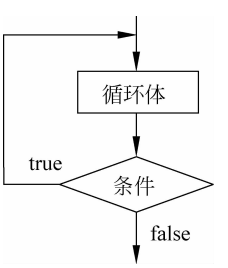

图 3-4 do-while 循环结构

该循环首先执行循环体,然后计算条件表达式。如果表达式的值为 true,则返回到循环的开始继续执行循环体,直到 termination 的值为 false 循环结束。这种循环一般称为“直到型”循环。该循环结构与 while 循环结构的不同之处是,do-while 循环至少执行一次循环体。

编写程序,要求用户从键盘上输入若干个 double 型数(输入 0 则结束),程序计算并输出这些数的总和与平均值。

程序 3.8 DoWhileDemo.java

```
import java.util.Scanner;
public class DoWhileDemo {
  public static void main(String[] args) {
    double sum = 0,avg = 0;
    int n = 0;
    double number;
    Scanner input = new Scanner(System.in);
```

```
    do{
      System.out.print("Enter a number: ");
      number = input.nextDouble();
      if(number != 0){
        sum = sum + number;
        n = n + 1;
      }
    }while(number!= 0);
    avg = sum / n;
    System.out.println("sum = " + sum);
    System.out.println("avg = " + avg);
  }
}
```

3.2.3 for 循环结构

for 循环是 Java 语言中 4 种循环结构中功能最强,也是使用最广泛的循环结构。它的一般格式如下:

```
for (initialization; termination; iteration){
  // 循环体
}
```

在 for 循环中,initialization 为循环的初始化部分,termination 为循环的条件,iteration 为迭代部分,三部分需用分号隔开。for 循环开始执行时首先执行初始化部分,该部分在整个循环中只执行一次。在这里可以定义循环变量并赋初值,可以定义多个循环变量,中间用逗号分隔,这里也是 Java 语言唯一可以使用逗号运算符的地方。

接下来判断循环的终止条件,若为 true 则执行循环体部分,否则退出循环。当循环体执行结束后,程序控制返回到迭代部分,执行迭代,然后再次判断终止条件,若为 true 则反复执行循环体。

在初始化部分可以声明多个变量,它们的作用域在循环体内,如下面循环中声明了 i 和 j 两个变量。

```
for(int i = 0, j = 10 ; i < j ; i++, j--) {
  System.out.println("i = " + i + " ,j = " + j);
}
```

for 循环中的一部分或全部可为空,循环体也可为空,但分号不能省略。

例如:

```
for ( ;  ; ){}
```

for 循环和 while 循环及 do-while 循环有时可相互转换。例如,有下面的 for 循环:

```
for(int i = 0, j = 10 ; i < j ; i++, j--){
    System.out.println("i = " + i + " ,j = " + j);
}
```

可以转换为下面等价的 while 循环结构。

```
int i = 0, j = 10;
while(i < j){
  System.out.println("i = "+ i + " ,j = " + j) ;
  i++;
  j-- ;
}
```

提示：在 Java 5 中增加了一种新的循环结构，称为增强的 for 循环，主要用于对数组和集合对象的元素迭代。关于增强的 for 循环在 5.1.2 节中讨论。

3.2.4 循环结构的嵌套

在一个循环结构的循环体中可以嵌套另一个完整的循环结构，称为**循环的嵌套**。内嵌的循环还可以嵌套循环，这就是多层循环。同样，在循环体中也可以嵌套另一个选择结构。

下面程序打印输出九九乘法表，这里使用了循环的嵌套。

程序 3.9 NineTable.java

```
public class Ninetable{
  public static void main(String[] args){
    int i , j;
    for(i = 1; i <= 9; i++){
        for(j = 1; j <= i; j++)
            System.out.print(j +" * " + i +" = " + i*j + "  ");
        System.out.println();
    }
  }
}
```

程序输出结果为：

```
1*1=1
1*2=2  2*2=4
1*3=3  2*3=6  3*3=9
1*4=4  2*4=8  3*4=12  4*4=16
1*5=5  2*5=10  3*5=15  4*5=20  5*5=25
1*6=6  2*6=12  3*6=18  4*6=24  5*6=30  6*6=36
1*7=7  2*7=14  3*7=21  4*7=28  5*7=35  6*7=42  7*7=49
1*8=8  2*8=16  3*8=24  4*8=32  5*8=40  6*8=48  7*8=56  8*8=64
1*9=9  2*9=18  3*9=27  4*9=36  5*9=45  6*9=54  7*9=63  8*9=72  9*9=81
```

3.2.5 break 语句和 continue 语句

在 Java 循环体中可以使用 continue 语句和 break 语句。

1. break 语句

break 语句是用来跳出 while、do、for 或 switch 结构的执行，该语句有两种格式：

```
break;
break lable;
```

break 语句的功能是结束本次循环,控制转到其所在循环的后面执行。对各种循环均直接退出,不再计算循环控制表达式。下面程序演示了 break 语句的使用。

程序 3.10　BreakDemo.java

```
public class BreakDemo{
  public static void main(String[] args){
    int n = 1;
    int sum = 0;
    while(n <= 100){
      sum = sum + n;
      if(sum > 100){
        break;  // 若条件成立退出循环
      }
      n = n + 2;
    }
    System.out.println("n = " + n);
    System.out.println("sum = " + sum);
  }
}
```

程序输出结果为:

```
n = 21
sum = 121
```

使用 break 语句只能跳出当前的循环体。如果程序使用了多重循环,又需要从内层循环跳出或者从某个循环开始重新执行,此时可以使用带标签的 break。

考虑下面代码:

```
start:
for(int i = 0; i < 3; i++){
  for(int j = 0; j <4; j++){
      if(j == 2){
        break start;}
      System.out.println(i +": " + j);
  }
}
```

这里,标签 start 用来标识外层的 for 循环,因此语句 break start;跳出了外层循环。上述代码的运行结果如下:

```
0 : 0
0 : 1
```

2. continue 语句

continue 语句与 break 语句类似,但它只终止执行当前的迭代,导致控制权从下一次迭代开始。该语句有下面两种格式:

```
continue;
continue label;
```

以下代码会输出 0～9 的数字，但不会输出 5。

```
for(int i = 0; i < 10; i++){
  if(i == 5){
    continue;
  }
  System.out.println(i);
}
```

当 i 等于 5 时，if 语句的表达式运算结果为 true，使得 continue 语句得以执行。因此，后面的输出语句不能执行，控制权从下一次循环处继续，即 i 等于 6 时。

continue 语句也可以带标签，用来标识从哪一层循环继续执行。下面是使用带标签的 continue 语句的例子。

```
start:
for(int i = 0; i < 3; i++){
  for(int j = 0; j < 4; j++){
      if(j == 2){
        continue start;
      }
      System.out.println(i + " : " + j);
  }
}
```

这段代码的运行结果如下：

```
0 : 0
0 : 1
1 : 0
1 : 1
2 : 0
2 : 1
```

注意：

(1) 带标签的 break 可用于循环结构和带标签的语句块，而带标签的 continue 只能用于循环结构。

(2) 标签命名遵循标识符的命名规则，不相互包含的块名字可相同。

(3) 带标签的 break 和 continue 语句不能跳转到不相关的标签块。

提示：在 C/C++语言中可以使用 goto 语句从内层循环跳到外层循环，但是在 Java 语言尽管将 goto 作为关键字，但不能使用，也没有意义。

下面的程序在循环体中使用了带标签的 break 语句和 continue 语句。

程序 3.11　LabelDemo.java

```
public class LabelDemo{
  public static void main(String[] args){
    outer:
    for(int i = 0; i < 3; i++){
      System.out.println("i = " + i);
      inner:
```

```
        for(int j = 0; j < 100; j++){
          if(j == 20){
            break outer;}
          if(j % 3 == 0){
            continue inner; }
          System.out.print(j + "  ");
        }
        System.out.println("This will not be print.");
      }
      System.out.print("\\n");
      System.out.println("Loop Finish");
    }
  }
```

该程序在执行内层循环时,当 j 的值为 20 时,执行 break outer;程序结束 outer 循环,若 j 能被 3 整除,返回到内层循环的迭代处继续执行。

程序输出结果为:

```
i = 0
1  2  4  5  7  8  10  11  13  14  16  17  19
Loop Finish
```

3.3 案 例 研 究

3.3.1 一位数加法练习程序

为一年级小学生编写一位数加法运算练习程序。程序开始运行随机生成两个一位数,让学生输入计算结果,程序给出结果是否正确。

程序 3.12 AdditionQuiz.java

```
import java.util.Scanner;

public class AdditionQuiz {
  public static void main(String[]args){
    int number1 = (int)(Math.random() * 10);
    int number2 = (int)(Math.random() * 10);
    Scanner input = new Scanner(System.in);
    System.out.print(number1 + " + " + number2 + " = ");
    int answer = input.nextInt();
    if(answer == number1 + number2){
      System.out.println("恭喜你,答对了!");
    }else{
      System.out.println("很遗憾,答错了!");
      System.out.println(number1 + " + " + number2 + " = " + (number1 + number2));
    }
  }
}
```

下面是程序的一次运行结果：

```
1 + 6 = 10
很遗憾,答错了!
1 + 6 = 7
```

3.3.2 任意抽取一张牌

从一副纸牌中任意抽取一张，并打印出抽取的是哪一张牌。已知，一副牌有 4 种花色，黑桃、红桃、方块和梅花。每种花色有 13 张牌，共有 52 张牌。可以将这 52 张牌编号，从 0～51。规定编号 0～12 为黑桃，13～25 为红桃，26～38 为方块，39～51 为梅花。

可以使用整数的除法运算来确定是哪一种花色，用求余数运算确定是哪一张牌。例如，假设抽出的数是 n，计算 n/13 的结果，若为 0，则牌的花色为黑桃；若为 1，则牌的花色为红桃；若为 2，则牌的花色为方块；若为 3，则牌的花色为梅花。计算 n%13 的结果可得到第几张牌。

程序 3.13　PickCards.java

```
public class PickCards {
  public static void main(String[] args){
   int card = (int) (Math.random() * 53);
   String suit = "",rank = "";
   switch(card / 13){     // 确定牌的花色
     case 0: suit = "黑桃";break;
     case 1: suit = "红桃";break;
     case 2: suit = "方块";break;
     case 3: suit = "梅花";break;
   }
   switch(card % 13){     // 确定是第几张牌
     case 0: rank = "A";break;
     case 10: rank = "J";break;
     case 11: rank = "Q";break;
     case 12: rank = "K";break;
     default: rank = "" + (card % 13 + 1);
   }
   System.out.println("你抽取的牌是: " + suit + " " + rank);
   }
}
```

下面是程序的一次运行结果：

```
你抽取的牌是: 梅花 2
```

3.3.3 求最大公约数

两个整数的最大公约数(greatest common divisor，GCD)是能够同时被两个数整除的最大整数。例如，4 和 2 的最大公约数是 2，16 和 24 的最大公约数是 8。

求两个整数的最大公约数有多种方法。一种方法是，假设求两个整数 m 和 n 的最大公

约数,显然 1 是一个公约数,但它可能不是最大的。可以依次检查 k(k=2,3,4,…)是否是 m 和 n 的最大公约数,直到 k 大于 m 或 n 为止。

程序 3.14　GCD.java

```
import java.util.Scanner;

public class GCD{
  public static void main(String[] args){
    Scanner input = new Scanner(System.in);
    System.out.print("Enter first integer: ");
    int m = input.nextInt();
    System.out.print("Enter second integer: ");
    int n = input.nextInt();

    int gcd = 1;
    int k = 2;
    while(k<=m && k<=n){
      if(m%k == 0 && n%k == 0) // 判断 k 是否能同时被 n1 和 n2 整除
        gcd = k;
      k++;
    }
    System.out.println("The GCD of " + m + " and " + n + " is " + gcd);
  }
}
```

下面是程序的一次运行结果:

```
Enter first integer: 16
Enter second integer: 24
The GCD of 16 and 24 is 8
```

计算两个整数 m 与 n 的最大公约数还有一个更有效的方法,称为辗转相除法或称欧几里得算法,其基本步骤如下:计算 r = m%n,若 r == 0,则 n 是最大公约数。若 r != 0,执行 m = n, n = r,再次计算 r = m%n,直到 r==0 为止,最后一个 n 即为最大公约数。请读者自行编写程序实现上述算法。

3.3.4　打印输出若干素数

下列程序计算并输出前 50 个素数,每行输出 10 个。

程序 3.15　PrimeNumber.java

```
public class PrimeNumber{
  public static void main(String[] args){
    int count = 0;
    int number = 2;
    boolean isPrime;
    System.out.println("The first 50 primes are: \\n");
    while(count < 50){
      isPrime = true;
```

```
        for(int divisor = 2; divisor * divisor <= number; divisor ++){
          if(number % divisor == 0){
            isPrime = false;
            break;
          }
        }
        if(isPrime){
          count ++;
          if(count % 10 == 0)
            System.out.println(number);
          else
            System.out.print(number + " ");
        }
        number ++;
      }
    }
}
```

程序输出结果如下：

```
The first 50 primes are:

2  3  5  7  11  13  17  19  23  29
31  37  41  43  47  53  59  61  67  71
73  79  83  89  97  101  103  107  109  113
127  131  137  139  149  151  157  163  167  173
179  181  191  193  197  199  211  223  227  229
```

3.3.5 打印一年的日历

编写程序提示用户从键盘上输入一个年份(如 2013)和该年第一天是星期几，星期用数字表示，0 表示星期日，1 表示星期一等。程序打印该年的每月的月历，如下所示。

```
          January  2013
-----------------------------------------------
Sun   Mon   Tue   Wed   Thu   Fri   Sat
                1     2     3     4     5
 6     7     8     9    10    11    12
13    14    15    16    17    18    19
20    21    22    23    24    25    26
27    28    29    30    31
…
          December  2013
-----------------------------------------------
Sun   Mon   Tue   Wed   Thu   Fri   Sat
 1     2     3     4     5     6     7
 8     9    10    11    12    13    14
15    16    17    18    19    20    21
22    23    24    25    26    27    28
29    30    31
```

程序要求输入的年份用于计算是否是闰年(二月份有 29 天),程序用变量 startDay 记录 1 月 1 日是星期几,然后使用(startDay+daysOfMonth)%7 表达式计算其他月 1 日是星期几。

程序 3.16　PrintCalendar.java

```
import java.util.Scanner;

public class PrintCalendar {
  public static void main(String[] args){
    Scanner input = new Scanner(System.in);
    System.out.print("Enter full year(e.g.2010): ");
    int year = input.nextInt();
    System.out.print("Enter the day(0 Sunday)of first day: ");
    int startDay = input.nextInt();
    int month = 0;              // 存储月份
    for(month = 1; month < 13; month++){
      String monthName = "";
      switch(month){
         case 1: monthName = "January";break;
         case 2: monthName = "February";break;
         case 3: monthName = "March";break;
         case 4: monthName = "April";break;
         case 5: monthName = "May";break;
         case 6: monthName = "June";break;
         case 7: monthName = "July";break;
         case 8: monthName = "August";break;
         case 9: monthName = "September";break;
         case 10: monthName = "October";break;
         case 11: monthName = "November";break;
         case 12: monthName = "December";
        }
        int daysOfMonth = 0;  // 存储当前月的天数
        if(month == 1||month == 3||month == 5||month == 7
            ||month == 8||month == 10||month == 12)
          daysOfMonth = 31;
        if(month == 4||month == 6||month == 9||month == 11)
          daysOfMonth = 30;
        if(month == 2){
          if(year % 4 == 0&&year % 100!= 0||year % 400 == 0)
            daysOfMonth = 29;
          else
            daysOfMonth = 28;
        }
        System.out.println("          " + monthName +"  " + year);
        System.out.println("-----------------------------------------------");
        System.out.println(" Sun Mon Tue Wed Thu Fri Sat");
        int i;
        for(i = 0;i < startDay;i++)
          System.out.print("    ");
        for(i = 1; i <= daysOfMonth;i++){
          System.out.printf("%4d",i);
```

```
            if((startDay + i) % 7 == 0)
              System.out.println();
          }
          startDay = (startDay + daysOfMonth) % 7;
          System.out.println("\\n");
      }
    }
}
```

运行该程序输入年 2013,输入星期 2(2013 年 1 月 1 日是星期二)可以输出 2013 年每月日历。

3.4 小　　结

分支结构和循环结构是结构化程序设计的两种重要程序流程控制结构。分支结构可以通过 if-else 和 switch 实现,循环结构包括 while 循环、do-while 循环、for 循环和增强的 for 循环。分支结构和循环结构可以相互嵌套。

3.5 习　　题

1. 写出下面程序运行的结果。

```
public class Foo{
  public static void main(String[] args){
    int i = 1;
    int j = i++;
    if((i > ++j) && (i++ == j)){
      i += j;
    }
    System.out.println("i = " + i + ", j = " + j) ;
  }
}
```

2. 给定下面代码段,问变量 i 可使用(　　)3 种数据类型。

```
switch (i) {
  default:
    System.out.println("Hello");
}
```

A. char　　B. byte　　C. float　　D. double

E. Object　　F. enum

3. 给定下面程序段,输出结果为(　　)。

```
int i = 1, j = 0 ;
switch(i)  {
  case 2:   j += 6;
  case 4:   j += 1;
```

```
    default:   j += 2;
    case 0:   j += 4;
  }
  System.out.println(j);
```

A. 6　　B. 1　　C. 2　　D. 4

4. 下面的程序有错误：

```
public class IfWhileTest {
  public static void main (String [ ]args)  {
    int x = 1, y = 6;
    if(x = y)
      System.out.println("Equal ");
    else
      System.out.println("Not equal ");
    while (y--)  { x++; }
    System.out.println("x = " + x+ "  y = " + y);
  }
}
```

若使程序输出下面结果，应如何修改程序。

```
Not equal
x = 7  y =  -1
```

5. 下面程序段执行后，i 、j 的值分别为(　　)。

```
int i = 1, j = 10;
do{
  if(i++>--j)  continue;
}while(i < 5);
```

A. i = 6　j = 5　　B. i = 5　j = 5

C. i = 6　j = 4　　D. i = 5　j = 6

6. 下面程序输出 2～100 的所有素数，请填空。

```
public class PrimeDemo {
  public static void main(String[ ] args){
    int i = 0, j = 0;
    for(i = 2; i <= 100; i++){
      for(j = 2; j < i; j++){
       if(i % j == 0)
        _______
       }
     if(_______)
      System.out.print(i + "  ");
    }
  }
}
```

7. 下面程序的执行结果为(　　)。

```
public class FooBar{
  public static void main(String[] args){
    int i = 0, j = 5;
    tp: for(; ; i++){
      for(; ; --j)
        if(i > j)  break tp;
    }
    System.out.println("i = " + i + ",j = " + j);
  }
}
```

A. i = 1, j = −1　　B. i = 0, j = −1

C. i = 1, j = 4　　D. i = 0, j = 4

E. 在第 4 行产生编译错误

8. 下列程序的输出结果是(　　)。

```
public class Ternary{
  public static void main(String[] args){
    int a = 5;
    System.out.println("值为 - " + ((a < 5) ? 9.9 : 9));
  }
}
```

A. 值为−9　　B. 值为−5　　C. 发生编译错误　　D. 都不是

9. 编写程序,接受用户从键盘上输入 10 个整数,比较并输出其中的最大值和最小值。

10. 求解“鸡兔同笼问题”:鸡和兔在一个笼里,共有腿 100 条,头 40 个,问鸡兔各有几只?

11. 从键盘上输入一个百分制的成绩,输出五级制的成绩,如输入 85,输出“良好”,要求使用 switch 结构实现。

12. 编写程序,从键盘上输入一个整数,计算并输出该数的各位数字之和。

例如:

```
请输入一个整数: 8899123
各位数字之和为: 40
```

13. 假设大学的学费年增长率为 7.8%,编程计算多少年后学费翻一番?

14. 编写程序,求出所有的水仙花数。水仙花数是这样的三位数,它各位数字的立方和等于这个三位数本身,如 $371=3^3+7^3+1^3$,371 就是一个水仙花数。

15. 从键盘上输入两个整数,计算这两个数的最小公倍数和最大公约数并输出。

16. 编写程序,求出 1～1000 的所有完全数。完全数是其所有因子(包括 1 但不包括该数本身)的和等于该数。例如,$28=1+2+4+7+14$,28 就是一个完全数。

17. 编写程序读入一个整数,显示该整数的所有素数因子。例如,输入整数为 120,输出应为 2、2、2、3、5。

18. 编写程序,计算当 $n=10\,000,20\,000,\cdots,100\,000$ 时 π 的值。求 π 的近似值公式如下。

$$\pi = 4 \times \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \frac{1}{13} + \cdots + \frac{1}{2n-1} - \frac{1}{2n+1}\right)$$