第 3 章　结构化程序设计——设计图书 ISBN 校验器

3.1　图书 ISBN 校验器

国际标准书号(International Standard Book Number,ISBN)是专门为识别图书等文献而设计的国际编号。存在两种形式的 ISBN 编码：10 位 ISBN 编码与 13 位 ISBN 编码。

10 位 ISBN 编码的组成方式：组号-出版者号-书序号-校验码，例如 7-302-08599-4。组号是国家、地区、语言的代号；出版者号是出版社代号，由其隶属的国家或地区的 ISBN 中心分配；书序号由出版社定义其发行图书的编号；最末位的校验码能够校验出 ISBN 号是否正确。10 位 ISBN 编码的校验码满足如下公式：

$$C_{10}=11-\mathrm{MOD}\left(\sum_{i=1}^{9}(11-i)\times C_i,11\right)$$

C_1～C_{10}分别表示 1 位 ISBN 编码，其中，C_1～C_9 取值为 0～9 的数字，C_{10}可以取值字母 X 及数字 0～9，如果校验码 $C_1=10$，则用 X 表示。

13 位 ISBN 编码的组成方式：前缀码-组号-出版者号-书序号-校验码，例如 978-7302-08599-7。在 10 位编码的基础上增加了前缀码：978 或 979。校验码满足如下公式：

$$C_{13}=10-\mathrm{MOD}\left(\sum_{\substack{i\leqslant 12\\ \text{且i为奇数}}}C_i+3\times\sum_{\substack{i\leqslant 12\\ \text{且i为偶数}}}C_i,10\right)$$

如果计算得到 $C_{13}=10$，则令 $C_{13}=0$，以保证最末位的校验码取值为 0～9 的数字。

3.2　程序设计思路

依然采用 IPO 程序设计模式。

(1) 程序的输入(I)：以字符串的形式输入待检测的 ISBN 编码。

(2) 程序处理过程(P)：这部分包括程序的核心算法，需要依次完成以下操作：

① 需要获取输入的 ISBN 编码的长度，进而根据长度选择不同的公式计算校验码；

② 获取输入编码的第 i 位数字，即 C_i；

③ 根据公式计算累加和及余数。

(3) 程序的输出(O)：给出判断结果，即是否为合法的 ISBN 校验码。

3.3 关键技术

为了使程序具有良好的结构，使程序易于设计、易于理解，通常高级语言都会提供 3 种基本的流程控制结构：顺序结构、分支结构和循环结构。顺序结构按照语句的书写顺序执行程序；分支结构根据条件的取值选择性地执行代码分支；循环结构根据循环条件重复执行代码段。

3.3.1 顺序结构

顺序结构指计算机按照语句的编写次序执行，没有分支和跳转语句。常见的顺序结构包括：表达式语句、复合语句及空语句。

1. 表达式语句

在表达式的末尾加上分号即可构成表达式语句，例如：count＝1。

2. 复合语句

使用大括号把若干语句和声明组合到一起即可构成复合语句。

3. 空语句

仅由分号构成，不执行任何操作。

3.3.2 分支结构

Java 语言中常用的分支结构包括 if 语句及 switch 语句。

1. if 语句

if 语句根据逻辑表达式的取值决定程序的执行分支。if 语句有如下 3 种格式。
格式 1：单分支语句。

```
if(逻辑表达式){
    语句序列;
}
```

流程图如图 3-1 所示。
格式 2：双分支语句。

```
if(逻辑表达式){
    语句序列 1;
} else {
```

```
    语句序列 2;
}
```

流程图如图 3-2 所示。

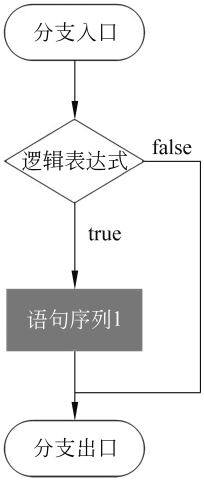

图 3-1　if 语句单分支流程图

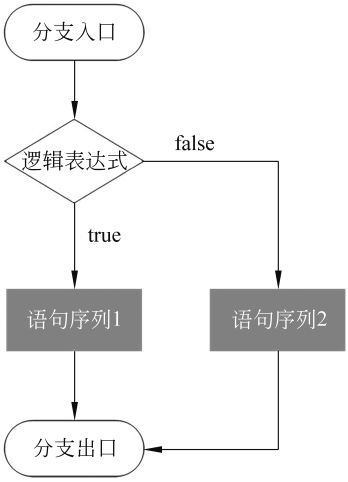

图 3-2　if 语句双分支流程图

格式 3：多分支语句。

```
if (逻辑表达式 1){
    语句序列 1;
} else if(逻辑表达式 2){
    语句序列 2;
} …
else {
    语句序列 n+1
}
```

流程图如图 3-3 所示。

3 种格式是相通的，如果格式 2 省略 else 子句，它就变成了格式 1；同理，格式 3 中省略 else if 子句就变会成格式 2。一般而言，如果语句序列中仅包含一行语句，则可以省略花括号；但是为了增强程序的可读性，通常不建议省略花括号。

分析下面的例 3-1，根据输入的成绩判断所属的成绩等级。

例 3-1

```
import java.util.Scanner;
public class ScoreGroup {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入成绩:");
        intscore = sc.nextInt();
        if(score>=0 &&score<=59) {
```

```
            System.out.println("不及格");
        }else if(score<=69) {          ①
            System.out.println("及格");
        }else if(score<=89) {
            System.out.println("良");
        }else if(score<=100) {
            System.out.println("优秀");
        }else {
            System.out.println("输入信息有误");
        }
        sc.close();
    }
}
```

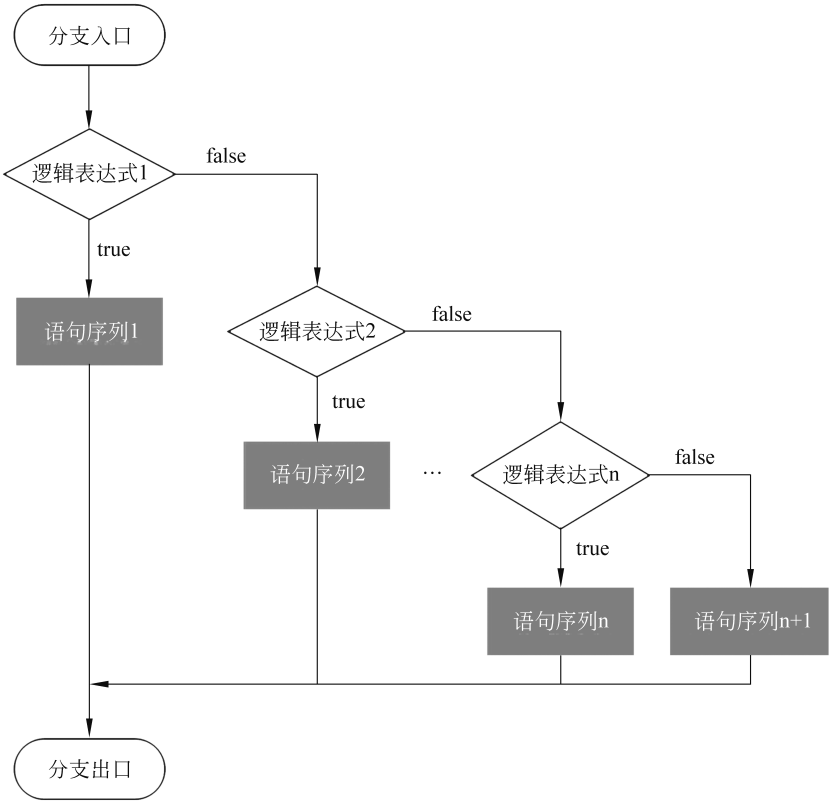

图 3-3　if 语句多分支流程图

表面上看起来，上面的程序没有问题，分别输入成绩 50、65、79、94、120 时，可以得到预期的输出结果。但是当输入成绩为－4 时，运行上面的程序，会得到及格的成绩等级，显然程序出现了问题。经过分析发现，主要原因在于 else 子句虽然后面仅包含一个条件，但是实际上 else 子句还隐含了一个条件：对前面条件取反。如例 3-1 的代码行①，条件是 score＜＝69，实际上这里隐含的条件是：!(score＞＝0 && score＜＝59)&& score

<=69,等价于 score<0 || (score>59 && score <=69)。显然,当输入成绩为-4 时,程序出现了问题。

为了得到正确的程序逻辑,把例 3-1 修改为如下形式。

```
import java.util.Scanner;
public class ScoreGroup {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("请输入成绩:");
        int score = sc.nextInt();
        if(score<0 || score>100) {
            System.out.println("输入信息有误");
        }else if(score>=90) {
            System.out.println("优秀");
        }else if(score>=70) {
            System.out.println("良");
        }else if(score>=60) {
            System.out.println("及格");
        }else {
            System.out.println("不及格");
        }
        sc.close();
    }
}
```

2. switch 语句

语法格式如下:

```
switch (表达式){
    case 值 1:语句序列 1;break;
    case 值 2:语句序列 2;break;
    …
    case 值 n:语句序列 n;break;
    default: 语句序列 n+1;
}
```

流程表示如图 3-4 所示。

switch 中的表达式只能是 byte、short、char、int 及枚举类型。Java 7 之后,增加了 String 类型的表达式。

switch 语句的执行过程:计算表达式将表达式的值按从上至下的顺序依次与 case 子句的常量进行比较。当表达式的值与 case 子句的常量相等时,执行其后的语句序列,直到遇到 break 或者 switch 语句执行结束。若没有与表达式的值相同的 case 常量,则执行 default 子句对应的语句序列,若程序省略 default 子句,则结束 switch 语句。

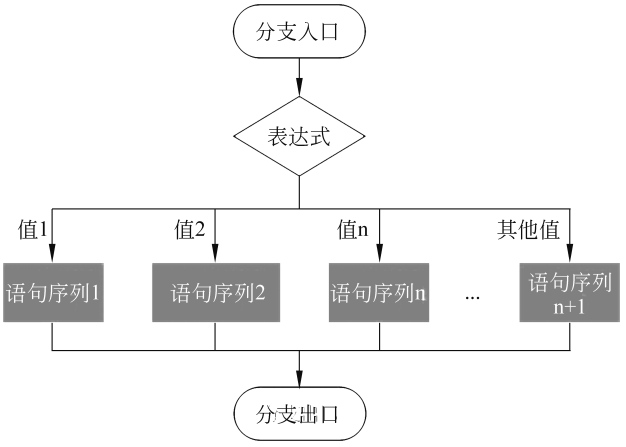

图 3-4　switch 语句多分支流程图

3.3.3　循环结构

Java 语言常用的 3 种循环结构为 while 循环、do-while 循环及 for 循环。

1. while 循环

语法格式如下：

```
初始化;
while(逻辑表达式){
    语句序列;
}
```

流程表示如图 3-5 所示。

while 循环的执行过程如下：

① 执行初始化操作；

② 计算逻辑表达式的值，若值为 true，则转到③，否则转到④；

③ 执行语句序列，转到②；

④ 循环出口，while 语句结束。

使用 while 循环时，一定要保证作为循环条件的逻辑表达式可以变成 false，否则这个循环就会成为死循环，程序无法结束。

2. do-while 循环

语法格式如下：

```
初始化;
do
{
```

```
    语句序列;
} while(逻辑表达式);
```

流程表示如图 3-6 所示。

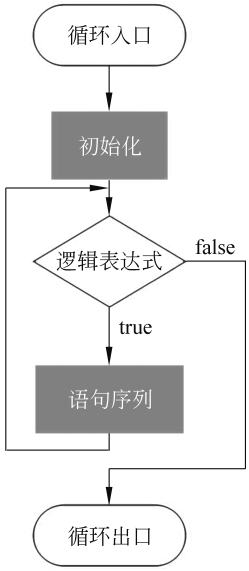

图 3-5　while 语句流程图

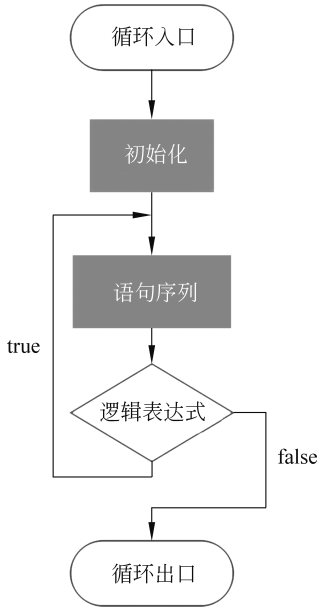

图 3-6　do-while 语句流程图

do-while 循环与 while 循环相比主要的区别在于：while 循环是先判断逻辑表达式后再执行循环语句序列；而 do-while 循环是先执行循环语句序列，然后判断逻辑表达式，如果逻辑表达式取值为 true，则继续执行语句序列，否则中止循环。

初学者使用 do-while 循环时容易出现的错误如下：

① 忽略 while(逻辑表达式)后的分号，造成编译错误；

② 在循环内部，声明循环变量，例如如下代码段。

```
do {
    System.out.println("请输入一个 1~10 之间的整数");
    int guess = sc.nextInt();
    if(guess<truth)
        System.out.println("太小了");
    else if(guess>truth)
        System.out.println("太大了");
}while(guess!=truth);
```

循环内部声明的局部变量 guess 的作用域仅限于大括号定界的语句序列，不包括 while 子句后面的逻辑表达式，所以上述程序会提示“cannot be resolved to a variable”错误。

3. for 循环

语法格式如下：

```
for(初始化;逻辑表达式;迭代表达式){
    语句序列;
}
```

流程表示如图 3-7 所示。

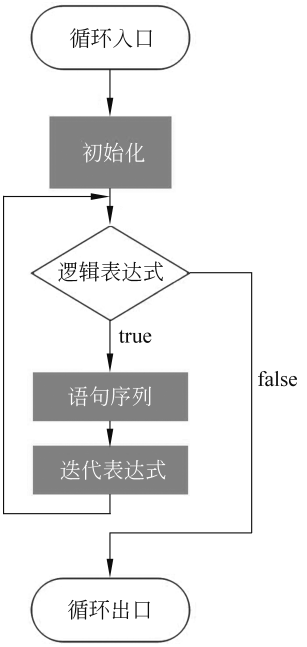

图 3-7　for 语句流程图

for 语句的执行过程如下：

① 执行初始化操作；

② 计算并判断逻辑表达式，若取值为 true，则转到③，否则转到⑤；

③ 执行语句序列；

④ 计算迭代表达式；

⑤ 结束 for 语句。

注意：在 for 循环结构中，初始化部分定义的局部变量的作用范围仅局限于 for 语句，即在 for 语句体之外，该局部变量将不可以使用。可以通过把初始化部分的代码放置在 for 语句之前扩大局部变量的作用范围。

3.3.4　循环控制结构

常用的循环控制结构包括 break 语句、continue 语句及 return 语句。

1. break 语句

break 语句用于结束循环。以 for 循环结构为例，语法定义如下：

```
for(初始化;逻辑表达式 1;迭代表达式){
    语句序列 1;
    if(逻辑表达式 2)break;
        语句序列 2;
}
```

流程控制如图 3-8 所示。

注意：对于循环嵌套结构，break 仅结束其所在的循环结构。

2. continue 语句

continue 语句的作用是忽略本次循环剩余的语句，继续执行下一次循环。以 for 循环结构为例，语法定义如下：

```
for(初始化;逻辑表达式 1;迭代表达式){
```

```
    语句序列 1;
    if(逻辑表达式 2)continue;
        语句序列 2;
}
```

流程控制如图 3-9 所示。

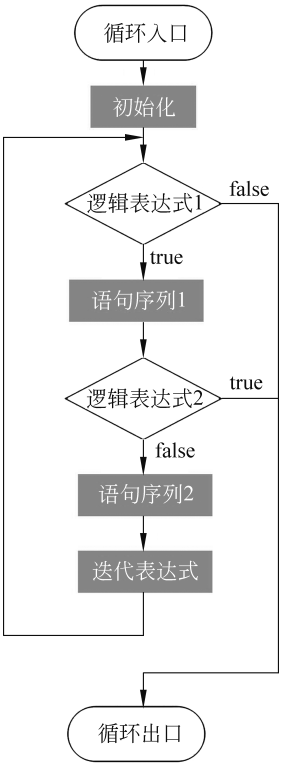

图 3-8　break 语句流程图

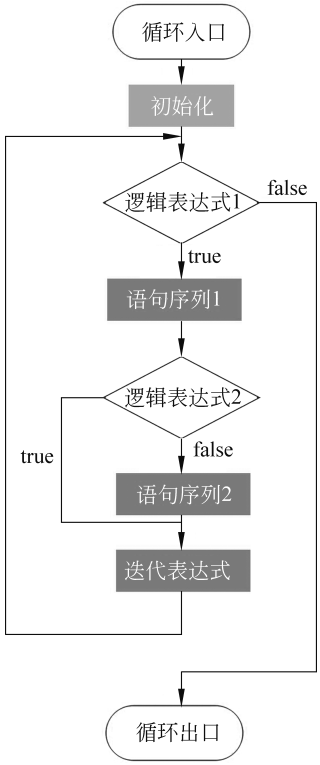

图 3-9　continue 语句流程图

注意：当逻辑表达式 2 取值为 true 时，流程控制将短路语句序列 2，会执行迭代表达式，继续下一次循环。

3. return 语句

结束当前方法的调用，返回主调方法。

3.4　图书 ISBN 校验器设计步骤

3.4

1. ISBN 编码的输入

利用 Scanner 类提供的 nextLine()方法可以实现字符串的输入。

2. digitAt()方法

digitAt()方法负责解析 ISBN 编码的每一位字符，并将其转换为对应的数字。首先利用字符串 String 类提供的 char charAt(int index)方法得到字符串的第 index 个字符，然后将其转换为数字。

```
public static int digitAt(String isbn, int index) {
    char digit = isbn.charAt(index);
    if(digit == 'X')
        return 10;
    else
        return digit - '0';
}
```

3. isISBN()方法

isISBN()方法判断读入的编码是否是合法的 ISBN 编码。

```
public static boolean isISBN(String isbn) {
    boolean result = false;
    int checkSum = 0;
    if (isbn.length() == 10) {
        for (int i= 1; i< 10; i++ )
            checkSum + = (11 - i) * digitAt(isbn, i - 1);
        checkSum = 11 - checkSum % 11;
        if (checkSum == digitAt(isbn, 9))
            result = true;
    } else if (isbn.length() == 13) {
        for (int i = 1; i < 13; i++ ) {
            if (i % 2 == 0)
                checkSum += 3 * digitAt(isbn, i - 1);
            else
                checkSum += digitAt(isbn, i - 1);
        }
        int remainder = checkSum % 10;
        if(remainder == 0)
            checkSum = 0;
        else
            checkSum = 10 - remainder;
        if (checkSum == digitAt(isbn, 12))
            result = true;
    }
    return result;
}
```

4. 定义主方法

定义主方法，调用上述方法，构建图书 ISBN 校验器。

```
public static void main(String[] args) {
        System.out.print("请输入图书的 ISBN 编码");
        Scanner sc = new Scanner(System.in);
        String isbn = sc.nextLine();
        if (isISBN(isbn))
            System.out.println("ISBN 校验码正确");
        else
            System.out.println("ISBN 校验码不正确");
        sc.close();
}
```

3.5 练一练

1. 按照如下个人所得税税率表及公式编写个人所得税计算器。

全月应纳税所得额＝扣除四金之后的应发工资－5000

提示：个人所得税的计算可以采用以下两种方法。

(1) 超额累积税率计算法如表 3-1 所示。

表 3-1 超额累积税率计算法

级数	全月应纳税所得额	税率/%
1	不超过 3000 元	3
2	超过 3000 元至 12000 元的部分	10
3	超过 12000 元至 25000 元的部分	20
4	超过 25000 元至 35000 元的部分	25
5	超过 35000 元至 55000 元的部分	30
6	超过 55000 元至 80000 元的部分	35
7	超过 80000 元的部分	45

(2) 速算扣除数法计算法如表 3-2 所示。

工资个税的计算公式为

应纳税额＝全月应纳税所得额×适用税率－速算扣除数

表 3-2 速算扣除数法计算法

级数	全月应纳税所得额	税率/%	速算扣除数
1	不超过 3000 元	3	0
2	3000 元至 12000 元	10	2520

续表

级数	全月应纳税所得额	税率/%	速算扣除数
3	12000 元至 25000 元	20	16920
4	25000 元至 35000 元	25	31920
5	35000 元至 55000 元	30	52920
6	55000 元至 80000 元	35	85920
7	超过 80000 元	45	181920

2. 通过键盘输入任意多个数字并计算其平均值。

提示：文本扫描器类 Scanner 的 hasNextDouble()可以判断是否存在下一个 double 类型的输入数据。

3. 用$\frac{\Pi}{4}\approx 1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\cdots$求 Π 的近似值，直到发现某一项的绝对值小于 10^{-6} 为止。

提示：对于“/”运算符，当两个操作数都为 int、long、short 时，计算结果只保留整数部分，舍弃小数部分。