第 3 章　Java 流程控制

流程控制是程序设计中的逻辑模块，内容包括顺序结构、选择结构、循环结构、跳转结构。这些结构通过程序语言的关键字定义，实现不同的功能。本章主要介绍 Java 程序设计的流程控制，并结合具体的实例加以说明。

3.1　语句与程序结构

3.1.1　语句

语句是编程语言中的基本结构，完成一定的功能。程序由一系列的语句组成，在语句后面加上分号表示一条语句的结束。通常语句有以下几种类型。

1. 空语句

空语句仅由一个分号组成，不执行任何操作，是语法的需要，具有特殊的功能。

2. 由表达式构成的语句

在表达式后面加上“;”即构成了表达式语句，表示表达式计算的结束。例如：

```
boolean a, b, c;                    //声明三个布尔类型的变量
a&b|c;                              //表达式语句
```

典型的赋值语句，如“boolean b=true;”，即为一条表达式语句。

3. 复合语句

用{}括起来的多个语句构成了复合语句，在语法上看如同一条语句。例如：

```
{
    C=A+B;
    C=C * C;
}
```

4. 方法调用

由方法调用构成的语句，如“System. out. println("a="+a));”语句实现了输出结果的功能。

5. 控制语句

实现程序的流程控制的语句，例如中断语句

```
break;
```

或跳转语句

```
continue LABEL;
```

3.1.2 程序结构

由语句构成的程序需要满足一定的逻辑结构才能实现问题的求解。这种结构描述了从数据入口到结果输出的整个流程,包含了顺序、循环、分支、跳转中的一个或多个过程。

3.2 顺序结构

顺序结构是程序设计中的最简单的结构,由赋值语句、输入输出语句等构成的结构,按照从左至右,从上至下的顺序执行。

【例 3-1】 顺序打印 3 个数。

程序代码如下:

```
1    public class Example3_1
2    {
3        public static void main(String[] args)
4        {
5            int a=1;                                    //声明 3 个整型变量
6            int b=2;
7            int c=3;
8            System.out.println("a+b="+(a+b));          //输出第 1 个和值
9            System.out.println("b+c="+(b+c));          //输出第 2 个和值
10           System.out.println("c+a="+(c+a));          //输出第 3 个和值
11       }
12   }
```

程序运行结果如下:

```
a+b=3
b+c=5
c+a=4
```

下例演示了顺序运算和输出的结果,结果输出的顺序和打印输出代码是一致的。

【例 3-2】 数值运算。

程序代码如下:

```
1    public class Example3_2
2    {
3        public static void main(String[] args)
4        {
5            int a=0, b=0;
6            float c=7.4f;
7            char d=' ', e='m';
8            d=e;
9            a=(int)c;
10           b=b+e;
11           System.out.print("a="+a);
```

```
12            System.out.print("\t");
13            System.out.print("b="+b);
14            System.out.print("\n");
15            System.out.print("d="+d);
16        }
17    }
```

程序输出结果如下：

```
a=7    b=109
d=m
```

3.3 选 择 结 构

根据某些条件进行选择在生活和生产中经常遇到，例如，在 ATM 取款机上需要选择查看的币种；交通信号灯的红、黄、绿；学生成绩的优、良、中、差。在编程语言中，选择性通过程序的选择结构来实现。选择结构也称为分支结构，分为单分支结构、双分支结构和多分支结构。

3.3.1 if 单分支结构

if 语句可以实现单分支结构，具有如下的格式：

```
if(条件表达式)
{
    语句块;
}
```

当“条件表达式”为真的时候，执行语句块，否则跳过。例如

```
⋮
int a=1, b=2, c=0;
if(a!=2 && b==3)
{
    c=5;
}
⋮
```

该程序段得到的 c 的值为 0，而不是 5。当分支结构内的程序语句只有一条时，对应的大括号可以省略。上面的 if 语句与下面的语句是等价的，

```
if(a!=2 && b==3)
c=5;
```

if 语句可以使用图 3-1 表示。当满足条件时执行语句块，否则，跳过语句块。

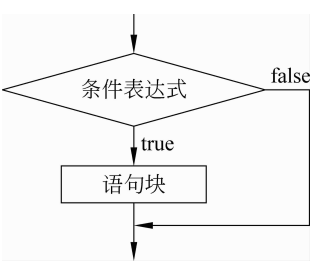

图 3-1 if 语句单分支结构

【例 3-3】 计算 3 个数中的最大数。

程序代码如下：

```
1    import java.util.*;
2    public class Example3_3
3    {
4        public static void main(String[] args)
5        {
6            int a=8, b=99, c=87;
7            if(a<b)
8                a=b;          //执行完毕后,a 的值为 a 和 b 中的最大值
9            if(a<c)
10               a=c;          //执行完毕后,a 的值为 a 和 c 中的最大值
11           System.out.println("maxValue="+a);
12         }
13   }
```

程序运行结果如下：

```
maxValue=99
```

程序分析如下：

程序返回结果 a、b、c 中的最大值。第一个 if 语句将 a、b 中的最大值赋给 a，然后在第二个赋值语句中将 a 和 c 的最大值赋给 c。

3.3.2 if 双分支结构

if…else 语句可以实现双分支结构，下面给出了基本的语法结构。

```
if(条件表达式)
{
    语句块 1;
}
else
{
    语句块 2;
}
```

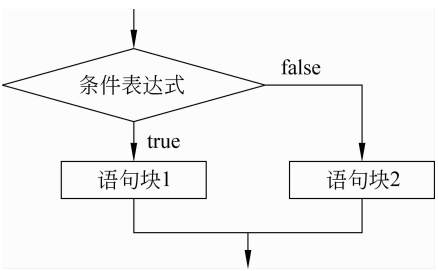

图 3-2 if 语句的双分支结构

图 3-2 表示了 if 语句的双分支结构。

【例 3-4】 根据 a，b 的大小决定输出。

程序代码如下：

```
1    import java.util.*;
2    public class Example3_4
3    {
4        public static void main(String[] args)
5        {
6            int a=8, b=99;
```

```
7            if(a<b)
8                System.out.println("a 小于 b");
9            else                //对 a>=b 的情况做处理
10               System.out.println("a 大于等于 b");
12           }
13   }
```

程序运行结果如下：

```
a 小于 b
```

3.3.3 if 多分支结构

if 语句的多分支结构(如图 3-3 所示)可以用以下的语法表达：

```
if(条件表达式 1)
{
    语句块 1;
}
else if(条件表达式 2)
{
    语句块 2;
}
⋮
else if(条件表达式 n)
{
    语句块 n;
}
[else
{
    语句块 n+1;
}]
```

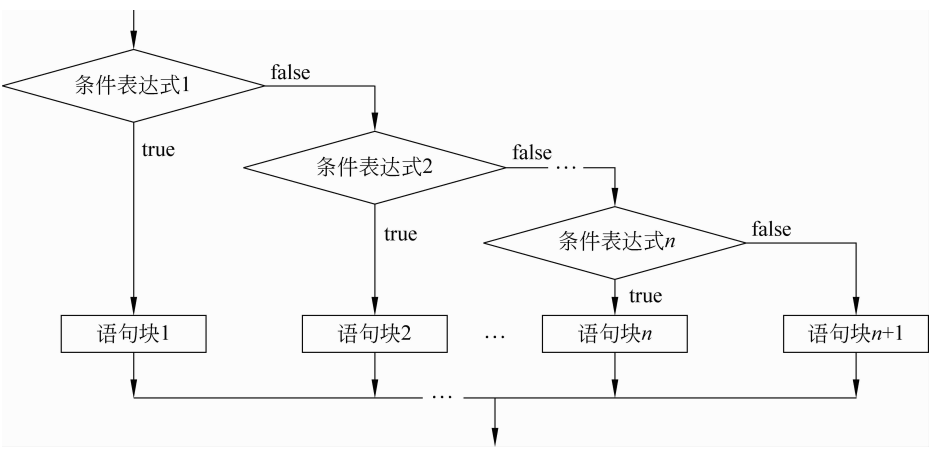

图 3-3　if 语句的多分支结构

【例 3-5】 根据输入的学生成绩来判定是否及格。

程序代码如下：

```
1    import java.io.*;                                          //调用系统函数
2    import java.util.Scanner;
3    public class Example3_5
4    {
5        public static void main(String[] args)
6        {
7            double stuGrade;                                   //学生成绩
8            System.out.println("请输入学生的成绩: ");          //界面提示
9            Scanner scanner=new Scanner(System.in);           //接收用户输入
10           String line=scanner.nextLine();
11           stuGrade=Double.parseDouble(line);                //将字符串转换为双精度型
12           if(stuGrade>=0 && stuGrade<60)                    //判断条件 1
13           {
14               System.out.println("不及格");
15           }
16           else if(stuGrade>=60 && stuGrade<=100)            //判断条件 2
17               System.out.println("及格");
18           else                                              //判断条件 3
19               System.out.println("输入错误,请重输:");
20       }
21   }
```

程序运行结果如下：

```
请输入学生的成绩:
55
不及格
请输入学生的成绩:
65
及格
```

程序分析如下：

程序运行后，将接收用户输入的数字串，如果数值位于 0～60 之间(不含 60)，则系统输出“不及格”，若位于 60～100 之间，则输出“及格”。若有其他不合适的输入，则系统要求用户重新输入。

3.3.4 if 的嵌套结构

if 语句可以嵌套，完成复杂的判断功能。if 语句的两重嵌套结构具有如下的形式：

```
if(条件表达式 1)
{
    ⋮
    if(条件表达式 2)
```

```
    {
        语句块;
    }
    ⋮
}
```

当然,除了两重嵌套,还可以有多重嵌套形式。下例利用 if 语句的两重嵌套实现了学生成绩的评价。

【例 3-6】 根据学生成绩给出评价。

程序代码如下:

```
1     import java.util.*;
2     import java.io.*;
3     public class Example3_6
4     {
5         public static void main(String[] args)
6         {
7             float grade=0.0f;
8             System.out.println("请输入分数: ");
9             Scanner input=new Scanner(System.in);        //输入学生成绩
10            grade=input.nextFloat();
11            if(grade<60)
12                System.out.println("不及格");
13            if(grade>=60&&grade<=100)
14            {
15                if(grade>=60&&grade<70)
16                    System.out.print("及格");
17                if(grade>=70&&grade<80)
18                    System.out.print("中等");
19                if(grade>=80&&grade<90)
20                    System.out.print("良好");
21                if(grade>=90&&grade<=100)
22                    System.out.print("优秀");
23            }
24        }
25    }
```

程序两次运行结果如下:

```
请输入分数:
87
良好
```

程序分析如下:

第 11～12 行是一个 if 语句。第 13 到 22 行实现了 if 语句的两重嵌套,在成绩不低于 60 分的情况下区分出了“及格”、“中等”、“良好”、“优秀”这 4 个级别。

3.3.5 switch 语句

当分支情况比较多的时候，可以采用 switch 语句有效避免判断的多层嵌套。switch 语句的语法结构如下：

```
switch(表达式)
{
    case 常量 1:
         语句块 1;
         break [标号];
    case 常量 2:
         语句块 2;
         break [标号];
    ⋮
    case 常量 n:
         语句块 n;
         [break [标号];
    [default:
         语句块 n+1;
         break;]
}
```

常量可以是变量或者表达式的取值，如{1,2,3}或者{'红','绿','蓝'}中的值，但是不能为字符串。如果在某个 case 语句中未写 break 语句，则程序将继续向下执行，不能跳出 switch 语句，直到遇到下一个 break 语句或者分支结束。当表达式未能匹配上 n 个常量中的任何一个时，系统默认执行 default 分支的语句块 $n+1$。default 分支是可选项，它总是位于 switch 语句最后。

图 3-4 给出了 switch 语句的分支结构图。

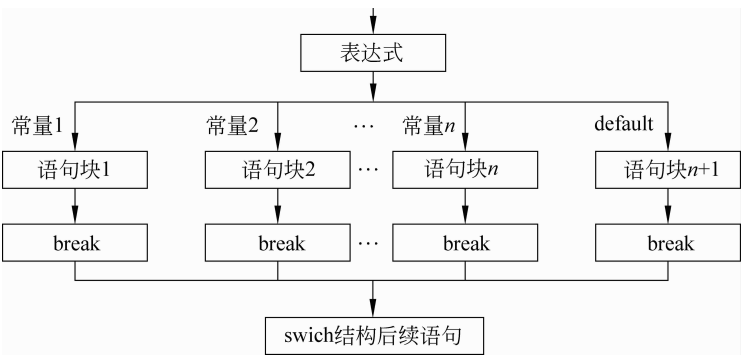

图 3-4 switch 语句的多分支结构

分支结构中，switch 中的表达式可以取不同类型的值，除了 Example3_4 中的整数值外，还可以取字符等。

【例 3-7】 ATM(自动取款机)根据用户的选择，执行不同的操作。

程序代码如下：

```
1    import java.util.*;
2    import java.io.*;
3    public class Example3_7
4    {
5        public static void main(String[] args)
6        {
7            int biZhong;                                    //币种
8            System.out.println("请选择币种：1人民币；2美元；3港币");
                                                            /*界面提示*/
9            Scanner scanner=new Scanner(System.in);    //接收用户输入的数字字符串
10           biZhong=scanner.nextInt();
11           switch(biZhong)                              //根据biZhong的不同取值分支
12           {
13               case 1:                                  //分支情况1
14                   System.out.println("您选择了人民币");
15                   break;                                //跳出switch结构,向下执行
16               case 2:                                  //分支情况2
17                   System.out.println("您选择了美元");
18                   break;
19               case 3:                                  //分支情况3
20                   System.out.println("您选择了港元");
21                   break;
22               default:                                 //系统默认
23                   System.out.println("您选择了人民币");
24                   break;
25           }
26       }
27   }
```

代码中定义了整型变量 biZhong 作为判别变量，使用 Scanner 对象接收用户的输入。nextInt 方法接收整数。在每个分支语句中，一定要用 break 语句结束。否则不能跳出 switch 结构，程序继续执行后继的分支。两次运行 Example3_5，结果分别如下：

```
请选择币种：1.人民币；2美元；3港币
1
您选择了人民币
-------------------------------------------------
请选择币种：1.人民币；2美元；3港币
3
您选择了港币
```

程序分析如下：

程序使用了整型变量作为分支的表达式。如果分支程序中忘记书写 break 语句，则不能得到正确的分支结果。将 Example3_5 中第 15 和 18 行的 break 去掉，运行程序，选择“1”得到如下结果：

```
请选择币种：1 人民币；2 美元；3 港币
1
您选择了人民币
您选择了美元
您选择了港元
```

显然，结果出现了错误。

switch 语句中不同的 case 情况可以对应一组语句，下面给出了一个例子。

【例 3-8】 选择不同类型的图像。

程序代码如下：

```
1     import java.util.*;
2     import java.io.*;
3     public class Example3_8
4     {
5         public static void main(String[] args)
6         {
7             char imageType;                                          //图像类型
8             System.out.println("请选择：y 灰度图像；r,g,b 彩色图像");/*界面提示*/
9             Scanner scanner=new Scanner(System.in);    //接收用户输入的数字字符串
10            imageType=scanner.nextLine().charAt(0);
11            switch(imageType)
12            {
13                case 'y':
14                    System.out.println("灰度图像");
15                    break;
16                case 'r':
17                case 'g':
18                case 'b':
19                    System.out.println("彩色图像");
20                    break;
21                default:
22                    System.out.println("请选择颜色");
23                    break;
24            }
25        }
26    }
```

程序两次运行结果如下：

```
请选择：y 灰度图像；r,g,b 彩色图像
g
彩色图像
-----------------------------------------------
请选择：y 灰度图像；r,g,b 彩色图像
y
```