第 3 章　结构语句

本章知识点：流程控制语句是用来控制程序中各语句执行顺序的语句，是程序中基本却又非常关键的部分。流程控制语句可以把单个的语句组合成有意义的、能完成一定功能的小逻辑模块。最主要的流程控制方式是结构化程序设计中规定的顺序结构、分支结构（选择结构）和循环结构三种基本流程结构。

本章将指导读者掌握 Java 程序中的流程控制语句，包括这些语句的语法结构和使用中需注意的要点。

实验 3.1　if 条件语句应用

1. 实验目的

（1）复习从键盘输入数据的方法。

（2）学习流程控制中的 if 条件语句。

2. 实验要求

编写一个 Java 程序，在程序中从键盘输入 3 个整数，比较它们的大小，并输出最大的数。

3. 程序运行结果

程序运行结果如图 3.1 所示。

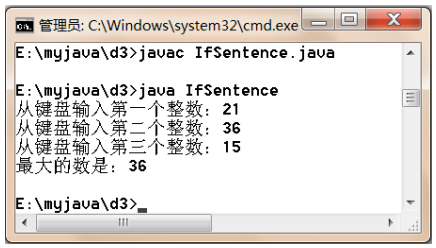

图 3.1　程序 IfSentence 运行结果

4. 程序模板

按模板要求，将代码 1～代码 5 替换为相应的 Java 程序代码，使之能输出图 3.1 所示的结果。

```
//文件名：IfSentence.java
```

```
import java.util.*;
public class IfSentence
{
  public static void main(String[] args)
  {
    int a=0,b=0,c=0,max=0;
    Scanner reader=new Scanner(System.in);
    System.out.print("从键盘输入第一个整数：");
    【代码 1】//从键盘输入一个整数并且赋值给变量 a
    System.out.print("从键盘输入第二个整数：");
    【代码 2】//从键盘输入一个整数并且赋值给变量 b
    System.out.print("从键盘输入第三个整数：");
    【代码 3】//从键盘输入一个整数并且赋值给变量 c
    【代码 4】//如果 a 大于 b，则 a 赋值给 max；否则，b 赋值给 max
    【代码 5】//如果 c 大于 max，则 c 赋值给 max
    System.out.println("最大的数是："+max);
  }
}
```

5. 实验指导

if 语句是 Java 程序中最常见的分支结构，它是一种“二选一”的控制结构，即给出两种可能的执行路径供选择。if 语句主要有双路条件选择和单路条件选择。

if 条件确定后，如果有多条语句要执行，要加上相应的大括号。其实，为了能更清晰地体现条件语句中的逻辑关系，即便只有一条语句，最好也要加上大括号。

实验 3.2　switch 语句及应用

1. 实验目的

（1）学习流程控制中 switch 语句的基本使用格式。

（2）掌握流程控制中 switch 语句的执行过程。

2. 实验要求

编写一个 Java 程序，让用户通过键盘输入一个 0~6 的整数，根据输入显示今天是星期几。

3. 程序运行结果

程序运行结果如图 3.2 所示。

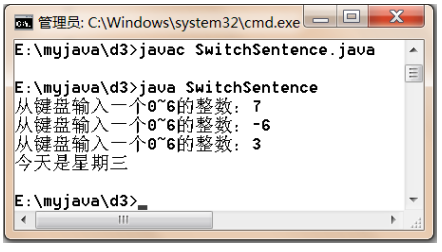

图 3.2　程序 SwitchSentence 运行结果

4. 程序模板

按模板要求，将代码 1、代码 9 替换为相应的 Java 程序代码，使之能输出图 3.2 所示的结果。

```
//文件名：SwitchSentence.java
import java.util.*;
public class SwitchSentence
{
  public static void main(String[] args)
  {
    int a= -1;
    Scanner reader=new Scanner(System.in);
    【代码 1】//从键盘输入 0~6 的整数，直到满足条件为止，并将其赋值给 a
    switch (a)
    {
      【代码 2】//当 a 为 0 时，显示今天是星期日
      【代码 3】//当 a 为 1 时，显示今天是星期一
      【代码 4】//当 a 为 2 时，显示今天是星期二
      【代码 5】//当 a 为 3 时，显示今天是星期三
      【代码 6】//当 a 为 4 时，显示今天是星期四
      【代码 7】//当 a 为 5 时，显示今天是星期五
      【代码 8】//当 a 为 6 时，显示今天是星期六
      【代码 9】//否则，显示输入错误
    }
  }
}
```

5. 实验指导

switch 语句是多分支的开关语句，常用于多重条件选择。它将一个表达式的值同许多其他值比较，并按比较结果选择下面将执行哪些语句。

在 switch 语句中，只有 int 型和 char 型可以作为 switch（表达式）中表达式的值。switch 语句的每一个 case 判断，在一般情况下都有一条 break 语句，以指明这个分支执行完后，就跳出该 switch 语句，如要若干判断值共享一个分支，则可以不写 break 语句，但需保留这些判断值中的最后一个 break 语句，这样可以实现由不同的判断语句流入相同的分支。

实验 3.3　for 循环语句与应用

1. 实验目的

（1）学习流程控制中 for 循环语句的基本使用格式。

（2）掌握流程控制中 for 循环语句的执行过程。

2. 实验要求

编写一个 Java 程序，用 for 循环语句求 1+3+5+…+99 的值。

3. 程序运行结果

程序运行结果如图 3.3 所示。

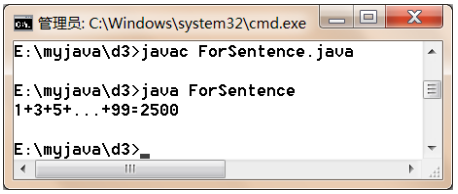

图 3.3　程序 ForSentence 运行结果

4. 程序模板

按模板要求，将代码 1～代码 3 替换为相应的 Java 程序代码，使之能输出图 3.3 所示的结果。

```
//文件名：ForSentence.java
public class ForSentence
{
  public static void main(String[] args)
  {
    int s=0;
    for (【代码 1】)  //声明一个变量 i，i 从 1 到 99，每次递增 1
    {
      【代码 2】        //s 和 i 相加，并将结果赋值给 s
      【代码 3】        //i 自增 1
    }
    System.out.println("1+3+5+…+99="+s);
  }
}
```

5. 实验指导

for 语句是 Java 语言 3 个循环语句中功能较强、使用较广泛的一个。在 for 循环中，如果在初始化表达式中定义了循环变量，那么这个变量的作用域范围是从循环开始到循环结束，如图 3.4 所示。

for(int j=1;j<5;j++)中定义的 j 的作用域只包括 for 循环内部，超出 for 循环，局部变量 j 的作用域就结束。所以，如果编译下面这个程序，会提示变量 j 未定义，如图 3.5 所示。

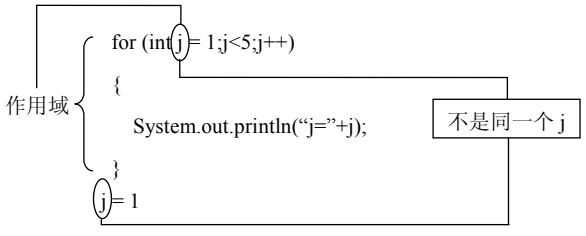

图 3.4　循环变量与局部变量的区别

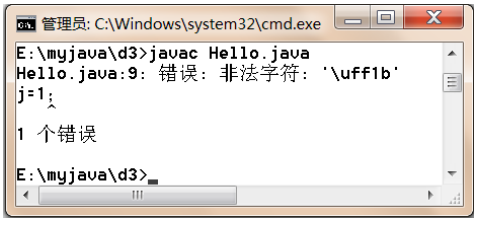

图 3.5　编译时的错误信息

```
public class Hello
{
  public static void main(String[] args)
```

```
  {
    for (int j=1;j<5;j++)
    {
      System.out.println("j="+j);
    }
    j=1;
  }
}
```

初学者由于对变量作用域的概念理解不透，普遍会出现这种错误，希望能引起大家的重视。

实验 3.4　while 循环语句与数据累加

1. 实验目的

（1）学习流程控制中 while 循环语句的基本使用格式。

（2）掌握流程控制中 while 循环语句的执行过程。

2. 实验要求

编写一个 Java 程序，在程序中从键盘输入一个范围在 50~100 的整数，如果不正确，则提示继续输入；然后求 1 到用户所输入整数的累加和。

3. 程序运行结果

程序运行结果如图 3.6 所示。

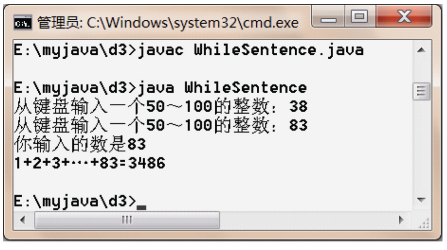

图 3.6　程序 WhileSentence 运行结果

4. 程序模板

按模板要求，将代码 1～代码 6 替换为相应的 Java 程序代码，使之能输出图 3.6 所示的结果。

```
//文件名：WhileSentence.java
import java.util.*;
public class WhileSentence
{
  public static void main(String[] args)
  {
    int a=0,i=1,s=0;
    Scanner reader=new Scanner(System.in);
    while （【代码 1】） //当 a<50 或者 a>100 时执行
```

```
    {
      【代码 2】//在屏幕上显示"从键盘输入一个 50～100 的整数："
      【代码 3】//读入一个整数，并且赋值给 a
    }
    System.out.println("你输入的数是"+a);
    while(【代码 4】) //当 i 不大于 a 时执行
    {
      【代码 5】  //将 i 累加到 s 上
      【代码 6】  //i 自增 1，考虑一下，如果 i 不改变值，循环会不会结束
    }
    System.out.println("1+2+3+…+"+a+"="+s);
  }
}
```

5. 实验指导

while 语句是循环语句，也是条件判断语句。while 语句的循环体可以是单个语句，也可以是复合语句块。while 语句的执行过程是先判断条件表达式的值，若为真，则执行循环体，循环体执行完之后，再转到条件表达式重新计算与判断条件表达式；直到当计算出的条件表达式为假时，跳过循环体执行 while 语句后面的语句，循环终止。

while 循环语句是在一定条件下，反复执行某段程序的控制结构。在实际使用中，应注意条件的改变，避免造成死循环。

实验 3.5　while 循环语句与字符比较

1. 实验目的

（1）学习利用 System.in.read()语句读取字符的方法。

（2）了解字符之间的比较使用“＝”运算符而不是使用 equals()方法。

2. 实验要求

利用 while 语句和 System.in.read()语句统计从键盘上所输入字符的个数。

3. 程序运行结果

程序运行结果如图 3.7 所示。

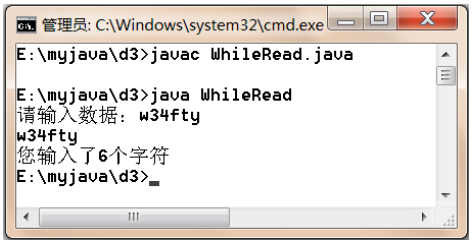

图 3.7　程序 WhileRead 运行结果

4. 程序模板

按模板要求，将代码替换为相应的 Java 程序代码，使之能输出图 3.7 所示的结果。

```
//文件名：WhileRead.java
```

```
import java.io.*;
public class WhileRead
{
  public static void main(String[] args) throws IOException
  {
    int count=0,b;
    System.out.print("请输入数据：");
    while(【代码】)  //调用 read()方法，读一个字符存入 b 中，按 Enter 键结束
    {
      System.out.print((char)b);        //输出字符 b
      count++;
    }
    System.out.print("\n 您输入了"+count+"个字符");
  }
}
```

5. 实验指导

利用 System.in.read()语句读取数据后，需要将其强制转换为 char 类型。还需注意的是，Enter 键包括回车符“\n”和换行符“\r”两个字符。字符的比较与字符串的比较是有区别的，字符的比较使用“=”运算符，而不能使用“==”运算符。

实验 3.6　do-while 循环语句

1. 实验目的

（1）学习流程控制中 do-while 循环语句的基本使用格式。

（2）掌握流程控制中 do-while 循环语句的执行过程。

2. 实验要求

编写一个 Java 程序，用 do-while 语句求 100+99+98+…+2+1 的值。

3. 程序运行结果

程序运行结果如图 3.8 所示。

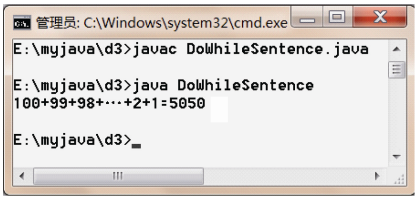

图 3.8　程序 DoWhileSentence 运行结果

4. 程序模板

按模板要求，将代码 1～代码 3 替换为相应的 Java 程序代码，使之能输出图 3.8 所示的结果。

```
//文件名：DoWhileSentence.java
public class DoWhileSentence
```

```
{
  public static void main(String[] args)
  {
    int i=100,s=0;
    do
    {
      【代码 1】    //将 i 累加到 s 上
      【代码 2】    //i 自减 1
    }
    while (【代码 3】);    //当 i 大于 0 时执行
    System.out.println("100+99+98+…+2+1="+s);
  }
}
```

5. 实验指导

do-while 语句的使用与 while 语句很类似，不同的是它不像 while 语句是先计算条件表达式的值，而是无条件地先执行一遍循环体，再来判断条件表达式的值，若表达式的值为真，则再执行循环体，否则跳出 do-while 循环去执行其后面的语句。do-while 语句的特点是它的循环体至少被执行一次。

do-while 语句的一般语法结构如下：

```
do
{
    循环体
}                          ——这个分号“;”一定不要丢
while (条件表达式);
```

实验 3.7　跳转语句

1. 实验目的

（1）学习流程控制中的 break 语句。

（2）学习流程控制中的 continue 语句。

2. 实验要求

编写一个 Java 程序，先显示 1～5 的各个整数；然后再显示 1～10 的各个奇数。

3. 程序运行结果

程序运行结果如图 3.9 所示。

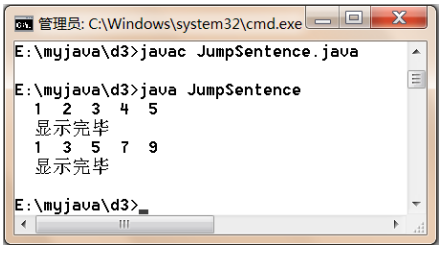

图 3.9　程序 JumpSentence 运行结果

4. 程序模板

按模板要求，将代码 1、代码 2 替换为相应的 Java 程序代码，使之能输出图 3.9 所示的结果。

```
//文件名：JumpSentence.java
public class JumpSentence
{
  public static void main(String[] args)
  {
    for(int i = 1; i < 10; i++)
    {
      if(i == 6)
        【代码 1】:    //跳出 for 循环
      System.out.print("  " + i);
    }
    System.out.println("\n  显示完毕");
    for(int i = 1; i < 10; i++)
    {
      if(i % 2 == 0)
        【代码 2】  //返回到 for 循环的头
      System.out.print("  " + i);
    }
    System.out.println("\n  显示完毕");
  }
}
```

5. 实验指导

break 语句将中断整个循环体，跳出循环，执行后续语句。continue 语句将结束本次循环，返回到循环开始处，开始新的一次循环。

第 3 章实验参考答案

实验 3.1：

```
【代码 1】: a=reader.nextInt();
【代码 2】: b=reader.nextInt();
【代码 3】: c=reader.nextInt();
【代码 4】: if (a>b)
             max=a;
           else
             max=b;
【代码 5】: if (c>max)
             max=c;
```

实验 3.2：

```
【代码 1】: while(a<0||a>6)
```

```
        {
          System.out.print("从键盘输入一个 0~6 的整数：");
          a=reader.nextInt();
        }
```

【代码 2】：`case 0:`

```
          System.out.println("今天是星期日");
          break;
```

【代码 3】：~【代码 8】略去

【代码 9】：`default:`

```
          System.out.println("输入错误");
```

实验 3.3：

【代码 1】：`int i=1;i<=99;i++`

【代码 2】：`s=s+i;`

【代码 3】：`i++;`

实验 3.4：

【代码 1】：`a<50||a>100`

【代码 2】：`System.out.print("从键盘输入一个 50～100 的整数：");`

【代码 3】：`a=reader.nextInt();`

【代码 4】：`!(i>a)`

【代码 5】：`s=s+i;`

【代码 6】：`i++;`

实验 3.5：

【代码】：`(char)(b=System.in.read())!='\r'`

实验 3.6：

【代码 1】：`s=s+i;`

【代码 2】：`i--;`

【代码 3】：`i>0`

实验 3.7：

【代码 1】：`break;`

【代码 2】：`continue;`