

第 3 章 结构语句

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。任何一门高级语言都定义了自己的流程控制语句来控制程序中的流程。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三种基本流程结构。

- 顺序结构
- 分支结构(选择结构)
- 循环结构

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 10 if 条件语句

1. 实验目的

- (1) 复习从键盘中输入数据的方法。
- (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三个整数,比较它们的大小,输出最大的数。

3. 程序运行效果

程序运行效果如图 3.1 所示。

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,使之能输出如图 3.1 所示的结果。

```
//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.println("从键盘输入第一个整数: ");
```

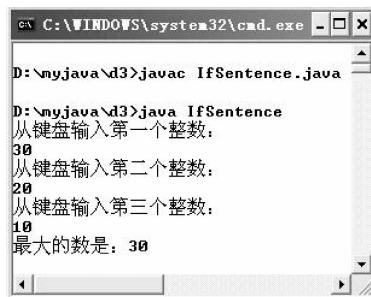


图 3.1 if 条件语句



```
    【代码 1】 //从键盘中输入一个整数并且赋值给变量 a
    System.out.println("从键盘输入第二个整数: ");
    【代码 2】 //从键盘中输入一个整数并且赋值给变量 b
    System.out.println("从键盘输入第三个整数: ");
    【代码 3】 //从键盘中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。

1) 双路条件选择

if-else 二路分支定义形式:

```
if (条件表达式)
{
    语句序列 1
}
else
{
    语句序列 2
}
```

2) 单路条件选择

if 单选定义形式:

```
if(条件表达式)
{
    语句序列
}
```

if 条件确定后, 如果有多条语句要执行, 要加上相应的花括号。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, 最好也要加上花括号。

实验 11 for 循环语句

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 $1+3+5+\cdots+99$ 的值。

3. 程序运行效果

程序运行效果如图 3.2 所示。



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,使之能输出如图 3.2 所示的结果。

```
//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);
    }
}
```



图 3.2 for 循环语句

5. 实验指导

for 语句是 Java 语言三个循环语句中功能较强,使用较广泛的一个。

for 循环语句的基本使用格式如下:

```
for(表达式 1; 条件表达式; 表达式 2)
{
    循环体
}
```

for 语句的执行过程是:首先计算表达式 1,完成必要的初始化工作;再判断条件表达式的值,若为真,则执行循环体,执行完循环体后再返回表达式 2,计算并修改循环条件,这样一轮循环就结束了。第二轮循环从计算并判断条件表达式开始,若表达式的值仍为真,则继续循环,否则,跳出整个 for 语句执行其后面的语句。

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 3.3 所示。

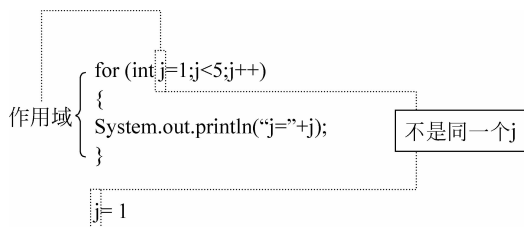


图 3.3 循环变量

for (int j=1; j<5; j++)中定义的 j 的作用域只包括 for 循环内部,若超出 for 循环,局部变量 j 作用域就结束。所以下面这个程序如果编译,会提示变量 j 未定义,如图 3.4 所示。



```
public class Hello
{
    public static void main(String[] args)
    {
        for (int j = 1; j < 5; j++)
        {
            System.out.println("j = " + j);
        }
        j = 1;
    }
}
```



图 3.4 编译错误

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, 希望读者能引起重视。

实验 12 while 循环语句

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, 在程序中通过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. 程序运行效果

程序运行效果如图 3.5 所示。

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, 使之能输出如图 3.5 所示的结果。



图 3.5 while 循环语句

```
//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 语句的一般语法结构如下:

```
while (条件表达式)
{
    循环体
}
```

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,再转向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 13 do-while 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 $100+99+98+\dots+2+1$ 的值。

3. 程序运行效果

程序运行效果如图 3.6 所示。

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,使之能输出如图 3.6 所示的结果。

```
//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);
    }
}
```



图 3.6 do-while 循环语句



```
}  
}
```

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,则再运行循环体,否则跳出 do-while 循环,执行下面的语句。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  
}  
while (条件表达式);
```

实验 14 switch 语句

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. 程序运行效果

程序运行效果如图 3.7 所示。

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,使之能输出如图 3.7 所示的结果。

```
//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】 //不断提示从键盘中输入一个 0~6 的整数,直到满足条件为止,并把输入的整数赋值给 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



图 3.7 switch 语句



```
        【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执行哪些语句。switch 选择语句的格式如下:

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

switch 语句的执行过程: 首先计算圆括号中“表达式”的值,这个值必须是整型或字符型的量,同时应与各个 case 后面的常量表达式值的类型相一致。计算出表达式的值后,将它先与第一个 case 后面的“常量表达式 1”的值相比较,若相同,则程序的流程转入第一个 case 分支的语句序列; 否则,再将表达式的值与第二个 case 后面的“常量表达式 2”相比较,以此类推; 如果表达式的值与任何一个 case 后的常量表达式值都不相同,则转去执行最后的 default 分支的语句序列。

在 switch 语句中,只有 int 型和 char 型可以作为 switch(表达式)中的表达式值。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。

下面的程序省略了 break 语句:

```
int a = 4;
switch (a)
{
    case 1:
        System.out.println("A");
```



```
case 2:
    System.out.println("B");
case 3:
    System.out.println("C");
case 4:
    System.out.println("D");
case 5:
    System.out.println("E");
case 6:
    System.out.println("F");
}
```

程序执行后会显示：

```
D
E
F
```

下面的程序省略了部分 break 语句：

```
int a = 3;
switch (a)
{
    case 1:
        System.out.println("A");
    case 2:
        System.out.println("B");
        break;
    case 3:
        System.out.println("C");
    case 4:
        System.out.println("D");
        break;
    case 5:
        System.out.println("E");
    case 6:
        System.out.println("F");
        break;
}
```

程序执行后会显示：

```
C
D
```

实验 15 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 各个整数;然后再显示 1~10 中的各个奇数。

3. 程序运行效果

程序运行效果如图 3.8 所示。

4. 程序模板

```
//JumpSentence.java
public class JumpSentence
{
    public static void main(String[] args)
    {
        for(int i = 1; i < 10; i++)
        {
            if(i == 6)
                break;
            System.out.print(" " + i);
        }
        System.out.println("\n 显示完毕");

        for(int i = 1; i < 10; i++)
        {
            if(i % 2 == 0)
                continue;
            System.out.print(" " + i);
        }
        System.out.println("\n 显示完毕");
    }
}
```

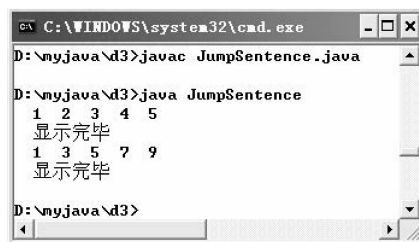


图 3.8 跳转语句

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。