

Java 程序设计

第 3 章 运算符、表达式和语句





导读

□ 主要内容

- ◆ 运算符与表达式
- ◆ 语句概述
- ◆ if \ switch 条件分支语句
- ◆ 循环语句
- ◆ break 和 continue 语句
- ◆ 数组与 for 语句

□ 重点和难点

- ◆ 重点：运算符、表达式和各种控制语句
- ◆ 难点：各种控制语句用法





3.1 运算符与表达式

- Java 提供了丰富的运算符，如算术运算符、关系运算符、逻辑运算符、位运算符等。
- Java 语言中的绝大多数运算符和 C 语言相同，基本语句，如条件分支语句、循环语句等也和 C 语言类似，因此，本章就主要知识点给予简单的介绍。



3.1.1

算术运算符与算术表达式

□ 加减运算符： $+$, $-$

◆ 加减运算符是二目运算符；加减运算符的结合方向是从左到右；

□ 乘、除和求余运算符： $*$ $/$ $\%$

◆ 以上运算符是二目运算符，结合方向是从左到右

□ 算术表达式

◆ 用算术符号和括号连接起来的符合 java 语法规则的式子，称为算术表达式

如： $x + 2 * y - 30 + 3 * (y + 5)$



3.1.2 自增，自减运算符

□ 自增、自减运算符： $++$, $--$

◆ 单目运算符，可以放在操作元之前，也可以放在操作元之后。操作元必须是一个整型或浮点型变量。作用是使变量的值增 1 或减 1，如：

◆ $++x$ ($--x$) 表示在**使用 x 之前**，先使 x 的值**增** (减) 1

◆ $x++$ ($x--$) 表示在**使用 x 之后**，使 x 的值**增** (减) 1

例如： `int f=2;`

`int m=2;`

`int x=(f * m++) + m;`

`int y=(f * ++m) + m;`

$x=7$

$x=9$



3.1.3 算术混合运算的精度

- 在计算算术表达式的值时，使用下列计算精度规则：
 - ◆ 表达式中有双精度浮点数（double 型数据），则按双精度进行运算。
 - ◆ 表达式中最高精度是单精度浮点数（float 型数据），则按单精度进行运算。
 - ◆ 表达式中最高精度是 long 型整数，则按 long 精度进行运算。
 - ◆ 表达式中最高精度低于 int 型整数，则按 int 精度进行运算。
 - ◆ char 型数据和整型数据运算结果的精度是 int。



3.1.3 算术混合运算的精度

例如： $5/2$ 的结果是 2，

要想得到 2.5，必须写成 $5.0/2$ 或 $5.0f/2$ 。

例如： `byte x=7;` 则执行表达式 `'B'+x;` 的结果是 `int` 型。



3.1.4 关系运算符与关系表达式

□ 关系运算符： $>$, $<$, $>=$, $<=$, $=$, \neq

◆ 运算规则：

- 在关系运算符中，当操作数是**基本数据类型**时，比较的是**数据内容**；
- 在关系运算符中，当操作数是**引用类型**时，比较的是引用对象的**引用值**，判断是否是同一对象，而没有比较对象的内容。



表 4.1 关系运算符

运算符	优先级	用法	含义	结合方向
>	6	op1>op2	大于	左到右
<	6	op1<op2	小于	左到右
>=	6	op1>=op2	大于等于	左到右
<=	6	op1<=op2	小于等于	左到右
==	7	op1==op2	等于	左到右
!=	7	op1!=op2	不等于	左到右

表 4.2 用逻辑运算符进行逻辑运算

op1	op2	op1&&op2	op1 op2	!op1
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true



3.1.5 逻辑运算符与逻辑表达式

□ 逻辑运算符包括： **&& 、 || 、 !**

- 其中： **&& 、 ||** 为二目运算符，实现逻辑与、逻辑或；
- **!** 为单目运算符，实现逻辑非。

□ 规则：

- 逻辑运算符的操作元必须是 **boolean** 型数据，逻辑运算符可以用来连接关系表达式，见表 4.2 。

□ **&& ， ||** 也称短路运算符



3.1.5

逻辑运算符与逻辑表达式

```
int i = 0;  
if (false & ++i == 1) {}  
System.out.println("i = " + i);
```

```
int i = 0;  
if (true | ++i == 1) {}  
System.out.println("i = " + i);
```

```
int j = 0;  
if (false && ++j == 1) {}  
System.out.println("j = " + j);
```

```
int j = 0;  
if (true || ++j == 1) {}  
System.out.println("j = " + j);
```

3.1.6

赋值运算符与赋值表达式

□ 赋值运算符： =

- ◆ 赋值运算符是二目运算符，左面的操作元必须是变量，不能是常量或表达式
- ◆ 赋值运算符的优先级较低，结合方向右到左
- ◆ 赋值表达式的值就是 “=” 左面变量的值

例如： `x = 20; b = true;`

注意：不要将赋值运算符 “=” 与等号逻辑运算符 “==” 混淆



3.1.7 位运算符

- **位运算符**：是指对两个整型数据按照对应的**位**进行运算，结果为新的整型数据。
- ◆ “按位与”运算 —— “&”是双目运算符
- ◆ “按位或”运算 —— “|”是双目运算符
- ◆ “按位非”运算 —— “~”是单目运算符
- ◆ “按位异或”运算 —— “^”是双目运算符
- ◆ 运算法则是：如果 a，b 两个数据对应位相同，则 c 的该位是 0，否则是 1。



例题

- 例子 3_1 中利用“异或”运算的性质，对几个字符进行加密并输出密文，然后再解密。

```
class Example3_1
{ public static void main(String args[])
{ char a1='十', a2='点', a3='进', a4='攻';
  char secret='8';
  a1=(char)(a1^secret);
  a2=(char)(a2^secret);
  a3=(char)(a3^secret);
  a4=(char)(a4^secret);
  System.out.println(" 密文 :"+a1+a2+a3+a4);
  a1=(char)(a1^secret);
  a2=(char)(a2^secret);
  a3=(char)(a3^secret);
  a4=(char)(a4^secret);
  System.out.println(" 原文 :"+a1+a2+a3+a4);
}
}
```



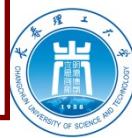


3.1.8 instanceof 运算符

□ instanceof 运算符

是二目运算符，左面的操作元是一个**对象**；右面是一个**类型**

```
class Test
{ public static void main (String argv[])
  { String s=new String("abc");
    if (s instanceof String)
      System.out.println("s is a string");
    B b=new B();
    if(b instanceof Object)
      System.out.println("b is definitely an object");
  }
}
```





3.1.9 运算符综述

- Java 的表达式就是用运算符连接起来的**符合 Java 规则**的式子。
- **运算符优先级**决定了表达式中运算执行的先后顺序。
- 编写程序时**尽量使用括号 () 运算符**来实现想要的运算次序，以免产生难以阅读或含糊不清的计算顺序
- 运算符的结合性决定了具有**相同级别运算符**的先后顺序



3.2 语句概述

□ Java 里的语句可分为以下六类

- ◆ **方法调用语句**：如： `System.out.println(" Hello");`
- ◆ **表达式语句**：表示式尾加上分号。如赋值语句： `x = 23;`
- ◆ **复合语句**：可以用 { } 把一些语句括起来构成复合语句，
如： `{ z = 123 + x;`
`System.out.println("How are you"); }`
- ◆ **空语句**：一个分号也是一条语句，称做空语句。
- ◆ **控制语句**：控制语句分为条件分支语句、开关语句和循环语句。
- ◆ **package 语句和 import 语句**：它们和类、对象有关，将在第 4 章讲解。





3.3 if 条件分支语句

□ 条件分支语句按着语法格式可分为三种形式：

◆ if 语句

◆ if-else 语句

◆ if-else if-- else if -else 语句



3.3.1 if 语句

□ if 语句是单条件分支语句，即根据一个条件来控制程序执行的流程（如图 3.2）。

◆ if 语句的语法格式：

```
if（表达式） {  
    若干语句  
}
```

注：表达式的值必须是 boolean 型的；不能用 0 代表 false；用 1 代表 true；

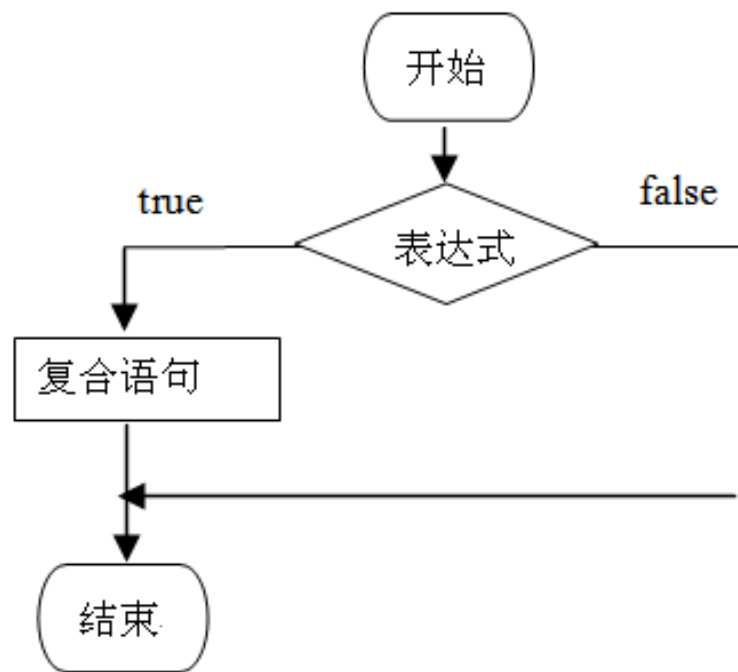


图 3.2 if 单条件、单分支语句

例题

例子 2 将变量

a , b , c
内存中的数值按大小顺序进行互换
(从小到大排列)。

```
public class Example3_2 {  
    public static void main(String args[]) {  
        int a = 9,b = 5,c = 7,t=0;  
        if(b<a) {  
            t = a; a = b; b = t;  
        }  
        if(c<a) {  
            t = a; a = c;c = t;  
        }  
        if(c<b) {  
            t = b; b = c; c = t;  
        }  
        System.out.println("a="+a+",b="+b+",c="+c);  
    }  
}
```



3.3.2 if-else 语句

□ if-else 语句是双条件分支语句，即根据一个条件来控制程序执行的流程。

◆ if-else 语句的语法格式：

```
if (表达式) {  
    若干语句  
}  
else {  
    若干语句  
}
```

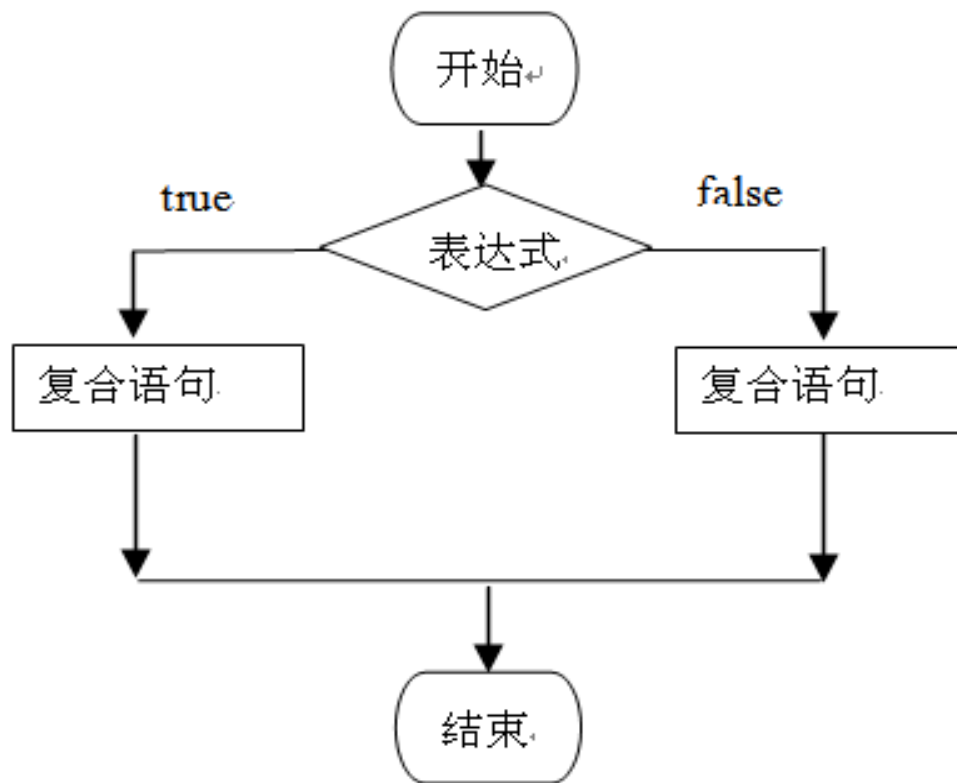


图 3.3 if-else 单条件、双分支语句





例题

- 例子 3 中有两条 if-else 语句，其作用是根据成绩输出相应的信息，运行效果如图 3.4 。

```
C:\chapter3>java Example3_3  
数学及格了  
英语不是优  
我在学习if-else语句
```

图 3.4 使用 if-else 语句

3.3.3 if-else if-else 语句

- if-elseif 语句是多条件分支语句，即根据多个条件来控制程序执行的流程。

- ◆ if-else if-else 语句的语法格式

```
if (表达式) {  
    若干语句  
}  
else if (表达式) {  
    若干语句  
}  
...  
else {  
    若干语句  
}
```

```
if (mark >= 90)  
    grade = "优";  
else if (mark >= 80)  
    grade = "良";  
else if (mark >= 70)  
    grade = "中";  
else if (mark >= 60)  
    grade = "及格";  
else  
    grade = "不及格";
```



多分支流程示意图

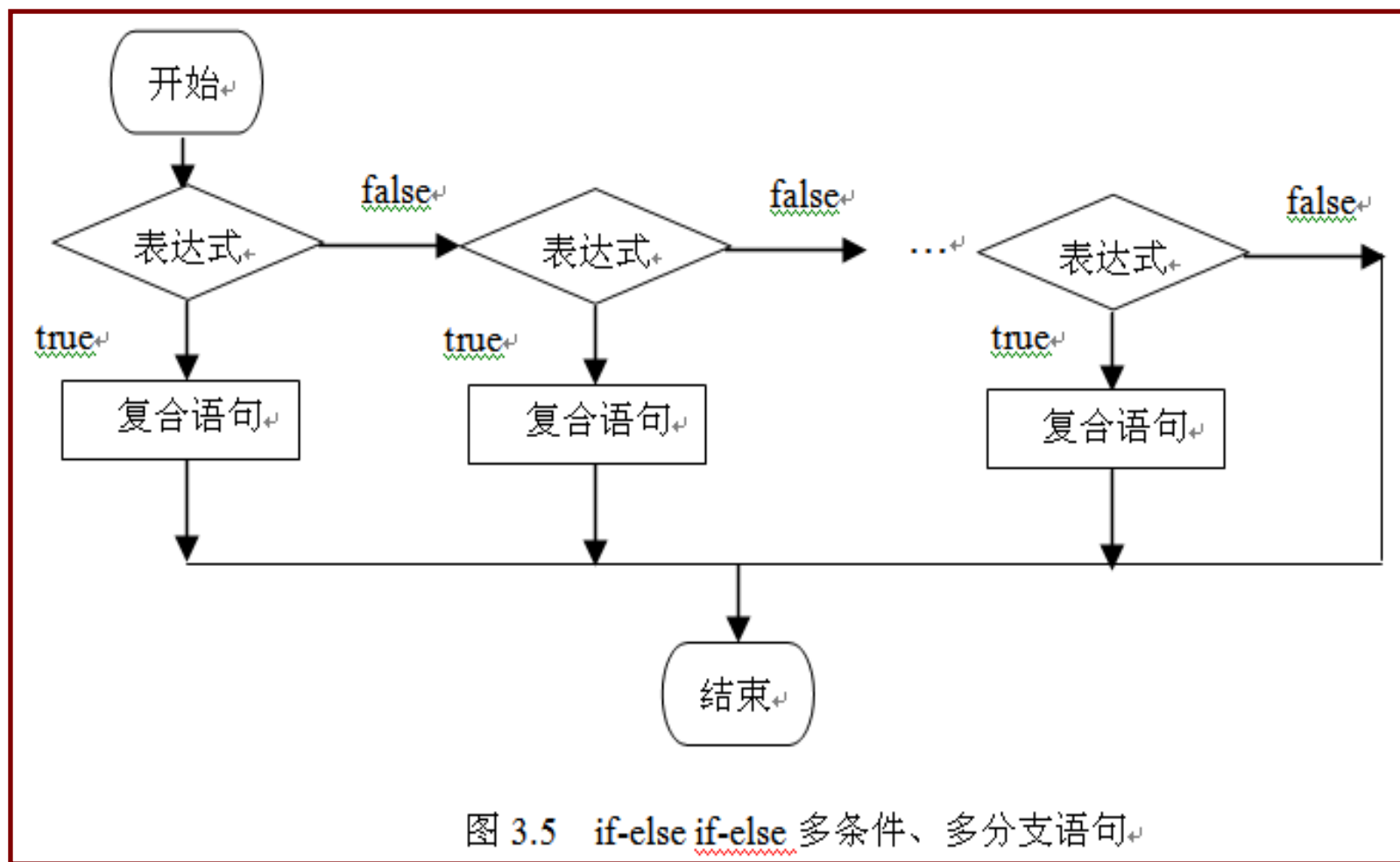


图 3.5 if-else if-else 多条件、多分支语句

3.4 switch 开关语句

□ switch 语句是单条件多分支的开关语句

```
switch( 表达式 )
```

```
{
```

```
    case 常量值 1:
```

若干个语

```
        break;
```

```
    case 常量值 2:
```

若干个语句

```
        break;
```

```
    ...
```

```
    case 常量值 n:
```

若干个语句

```
        break;
```

```
    default:
```

若干个语句

```
}
```

必须是**整型或字符型**的量，并且与各个 case 后面的常量表达式值的类型一致

每个 case 一般都有 break，若没有则多个 case 共享一个分支

表达式的值与任何一个 case 后的常量表达式的值不相同同时执行

- **例子 4** 使用了 switch 语句判断用户从键盘输入的正整数是否为中奖号码。

```
public class Example3_4 {  
    public static void main(String args[]) {  
        int number = 0;  
        System.out.println(" 输入正整数 ( 回车确定 )");  
        Scanner reader = new Scanner(System.in);  
        number = reader.nextInt();  
        switch(number) {  
            case 9 :  
            case 131 :  
            case 12 :    System.out.println(number+" 是三等奖" ); break;  
            case 209 :  
            case 596 :  
            case 27 :    System.out.println(number+" 是二等奖" ); break;  
            case 875 :  
            case 316 :  
            case 59 :    System.out.println(number+" 是一等奖" ); break;  
            default:    System.out.println(number+" 未中奖 ");  
        }  
    }  
}
```



```
switch (month) {  
    case 2:  
        days = 28;  
        break;  
    case 4:  
    case 6:  
    case 9:  
    case 11:  
        days = 30;  
        break;  
    default:  
        days = 31;  
        break;  
}
```

```
System.out.println(month + " 月份有 " + days + "
```





- **switch 多路选择**

- **break 是否可以不加？ ？ ？**
- **default 是否可以不加？ ？ ？**
- **表达式可以是字符串类型吗？ ？ ？**



3.5.1 for 循环语句

- **for 语句的语法规式：**

```
for ( 表达式 1; 表达式 2; 表达式 3) {  
    若干语句  
}
```

- **for 语句的执行规则是：**

- ① 计算“表达式 1”，完成必要的初始化工作
- ② 判断“表达式 2”的值，若“表达式 2”的值为 true，则进行 (3)，否则进行 (4)
- ③ 执行循环体，然后计算“表达式 3”，以便改变循环条件，进行 (2)
- ④ 结束 for 语句的执行。

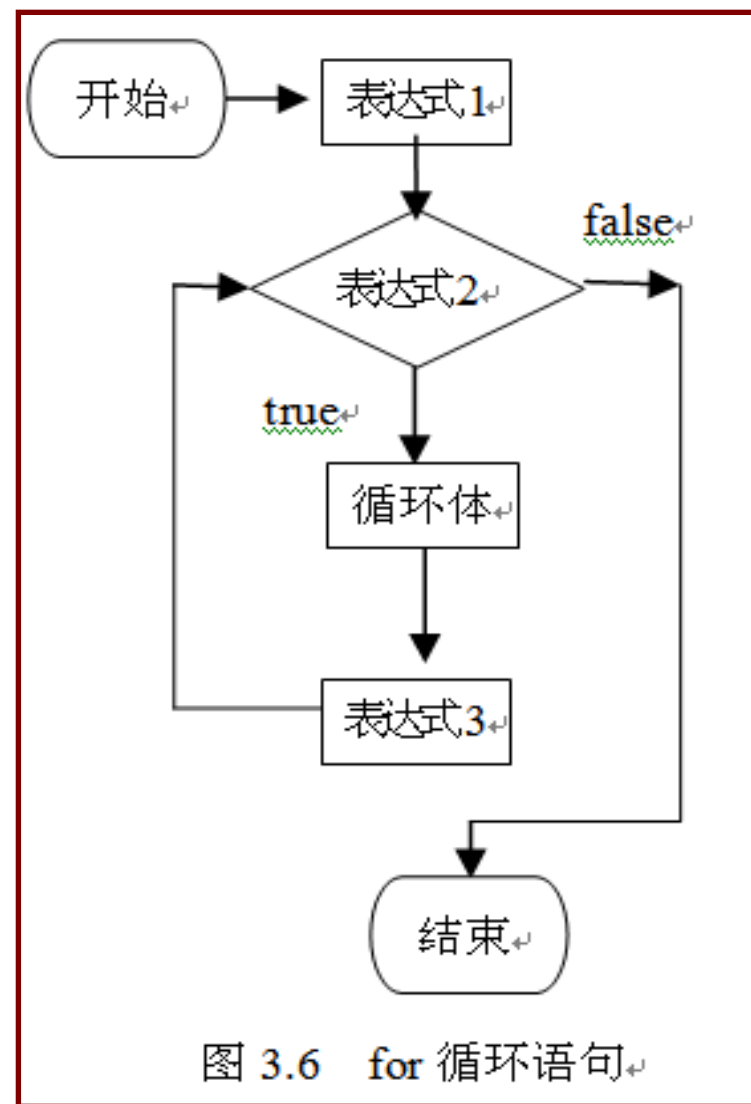


图 3.6 for 循环语句



例题

例子 5 : 计算 $8+88+888+8888\ldots$ 的前 12 项和。

```
public class Example3_5 {  
    public static void main(String args[]) {  
        long sum = 0, a = 8, item = a, n = 12, i = 1;  
        for(i=1; i<=n; i++) {  
            sum = sum+item;  
            item = item*10+a;  
        }  
        System.out.println(sum);  
    }  
}
```



3.5.2 while 循环

- **while 语句的语法格式:**

```
while ( 表达式 ) {  
    若干语句  
}
```

- **while 语句的执行规则是:**

- ① 计算表达式的值，如果该值是 true 则执行 (3)
- ② 执行循环体，再进行 (1)
- ③ 结束 while 语句的执行

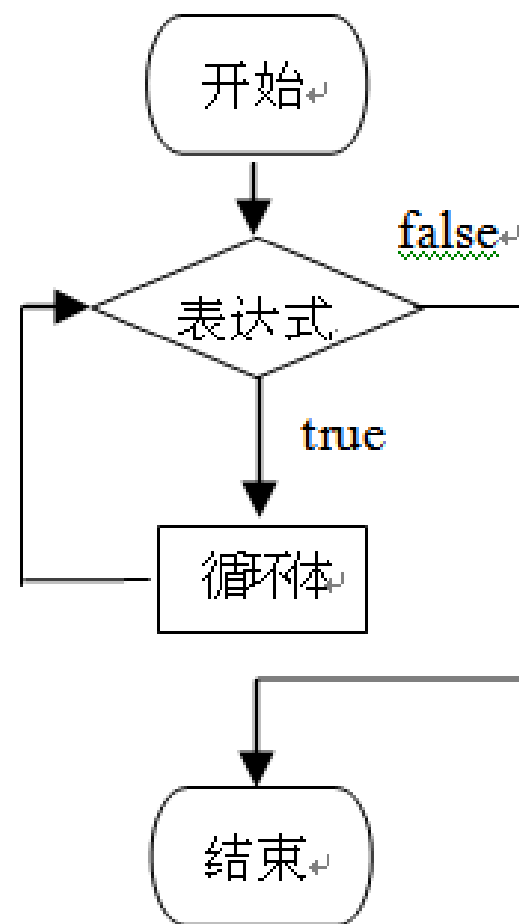



图 3.7 while 循环语句





```
int i, sum, sumOdd, sumEven;
i = 1; sum = 0;
sumOdd = 0;
sumEven = 0;
while (i <= 100)
{
    sum += i;
    if (i % 2 == 1)
        sumOdd += i;
    else if (i % 2 == 0)
        sumEven += i;
    i++;
}
System.out.println(" 总和:  " + sum);
System.out.println(" 奇数和:  " + sumOdd);
System.out.println(" 偶数和:  " + sumEven);
```



3.5.3 do-while 循环

- do-while 语句的语法格式:

```
do {  
    若干语句  
} while( 表达式 );
```

- do-while 语句的执行规则是:

- ① 执行循环体, 再进行 (2)
- ② 计算表达式的值, 如果该值是 true 就进行 (1), 否则执行 (3)
- ③ 结束 while 语句的执行。

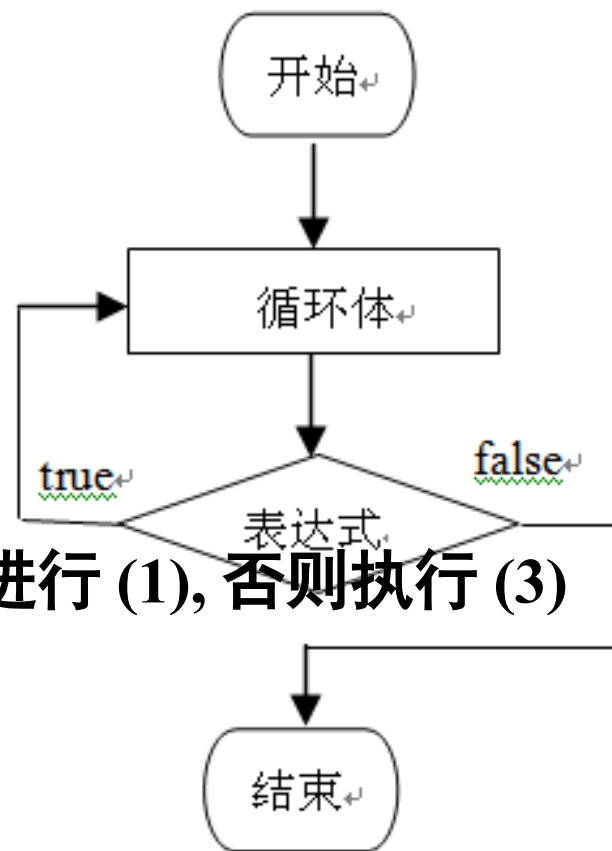



图 3.8 do-while 循环语句





```
int i, sum, sumOdd, sumEven;
i = 1; sum = 0;
sumOdd = 0;
sumEven = 0;
do {
    sum += i;
    if (i % 2 == 1)
        sumOdd += i;
    else if (i % 2 == 0)
        sumEven += i;
    i++;
}
while (i <= 100);
System.out.println(" 总和: " + sum);
System.out.println(" 奇数和: " + sumOdd);
System.out.println(" 偶数和: " + sumEven);
```



例题

- **例子 6** 用 while 语句计算 $1+1/2!+1/3!+1/4! \dots$ 的前 20 项

```
public class Example3_6 {  
    public static void main(String args[]) {  
        double sum = 0,item = 1;  
        int i = 1, n = 20;  
        while(i<=n) {  
            sum = sum+item;  
            i = i+1;  
            item = item*(1.0 / i);  
        }  
        System.out.println("sum="+sum);  
    }  
}
```



- 
- **猜数游戏。产生一随机整数，用户猜数，提示大小，直至猜到为止**



3.6

break 和 continue 语句

□ break 语句

◆ 使程序的流程从一个语句块（switch 或循环结构）内跳出。

◆ break [标号] ;

用于指定一个封闭的语句块，可以省略

□ continue 语句

◆ 终止当前这一轮（次）的循环，进入下一轮（次）循环。

◆ continue [标号] ;

应为定义在程序中某一循环语句的前面，用来标志该循环，它使得程序直接转入标号标明的循环层次。可以省略

□ return 语句

◆ 用来使程序从方法（函数）中返回，并返回一个值。

◆ return 返回值;



例题

- 例子 7 使用了 `continue` 和 `break` 语句，计算 $1+3+5\ldots$ 输出 100 以内的素数。

```
public class Example3_7 {  
    public static void main(String args[]) {  
        int sum=0,i,j;  
        for( i=1;i<=10;i++) {  
            if(i%2==0) {           // 计算 1+3+5+7+9  
                continue;  
            }  
            sum=sum+i;  
        }  
        System.out.println("sum="+sum);  
        for(j=2;j<=100;j++) {    // 求 100 以内的素数  
            for(i=2;i<=j/2;i++) {  
                if(j%i==0)  
                    break;  
            }  
            if (i>j/2) {  
                System.out.println(""+j+" 是素数 ");  
            }  
        }  
    }  
}
```



□break 语

句

```
for (int i = 0; i < 5; i++) {  
    for (int j = 0; j < 8; j++) {  
        if (j == 4)  
            break;  
        System.out.println(" 内层循环 ");  
    }  
    System.out.println(" 外层循环 ");  
}
```



□continue 语

句

```
for (int i = 0; i < 5; i++) {  
    for (int j = 0; j < 8; j++) {  
        if (j == 4)  
            continue;  
        System.out.println(" 内层循环 ");  
    }  
    System.out.println(" 外层循环 ");  
}
```




3.7 for 语句与数组

□ foreach 的语句

- 不用下标就可遍历整个数组
- foreach 语句需提供**元素类型**、**循环变量的名字**（用于存储连续的元素）和用于从中检索元素的**数组**

□ 语法：

```
for (type element : array)
{
    System.out.println(element);
    .....
}
```

例如：

```
int[ ] arr = {1,2,3,4,5};
for (int t : arr)
    System.out.println(t);
```



例题

- 例子 8 分别使用 for 语句的传统方式和改进方式遍历数组。

```
public class Example3_8 {  
    public static void main(String args[]) {  
        int a[] = {1,2,3,4};  
        char b[] = {'a','b','c','d'};  
        for(int n=0;n<a.length;n++) { // 传统方式  
            System.out.println(a[n]);  
        }  
        for(int n=0;n<b.length;n++) { // 传统方式  
            System.out.println(b[n]);  
        }  
        for(int i:a) { // 循环变量 i 依次取数组 a 的每一个元素的值 (改进方式)  
            System.out.println(i);  
        }  
        for(char ch:b) { // 循环变量 ch 依次取数组 b 的每一个元素的值 (改进方式)  
            System.out.println(ch);  
        }  
    }  
}
```



3.8 应用举例

- **例 9**，用户在键盘依次输入若干数字，每输入一个数字都需按回车确认，输入非数字字符串时结束整个输入操作过程。程序将计算出这些数的和以及平均值

```
Scanner reader=new Scanner(System.in);
```

```
double sum=0;
```

```
int m=0;
```

```
while(reader.hasNextDouble(){
```

```
    double x=reader.nextDouble();
```

```
    m=m+1;
```

```
    sum=sum+x;
```

```
}
```

```
System.out.printf("%"d 个数的和为 %f\n",m,sum);
```

```
System.out.printf("%"d 个数的平均值是 %f\n",m,sum/m);
```

```
98
129.77
865.88
end
3个数的和为1093.650000
3个数的平均值是364.550000
```

图 3.9 计算平均值





1 Java 提供了丰富的运算符，学会各种运算符的使用

2 Java 语言控制语句分为条件控制语句和循环控制语句

3 Java 提供了遍历数组的循环语句。

