

# Java语言程序设计

配套教材由清华大学出版社出版发行

## 第2章 Java语言基础



中國農業大學

阚道宏

# 第2章 Java语言基础

- Java语言的基础语法
  - 数据类型
  - 变量与常量
  - 运算符与表达式
  - 算法结构与控制语句
- Java语言与C/C++语言的基础语法



# 第2章 Java语言基础

- 本章内容
  - [2.1 数据类型](#)
  - [2.2 变量与常量](#)
  - [2.3 运算符与表达式](#)
  - [2.4 算法结构与控制语句](#)



## 2.1 数据类型

- 计算机中的数据存储
  - 存储位数
  - 存储格式
- 存储位数
  - 一个字节：  $(00000000)_2 \sim (11111111)_2$ ，即0~255
  - 定长存储
    - 8位（1字节）： 0 ~ 255
    - 16位（2字节）： 0 ~ 65535
    - 32位（4字节）： 0 ~ 4294967295
  - 编写程序时应根据所处理数据可能的取值范围合理地选择存储位数



## 2.1 数据类型

- 存储格式
  - 如何区分正数和负数、整数和实数？
- 如何区分正数和负数？
  - 将最高位作为符号位：0表示正数，1表示负数
  - 有符号格式、无符号格式

存储位数（字节数）	数值范围	
	无符号格式	有符号格式（补码）
8（1）	0 ~ 255	-128 ~ +127
16（2）	0 ~ 65535	-32768 ~ +32767
32（4）	0 ~ 4294967295	-2147483648 ~ 2147483647



# 2.1 数据类型

- 计算机如何存储一个实数呢？

- 实数的科学表示法

82.625, 8.2625, 0.82625, 0.082625

$0.82625 \times 10^2$ ,  $0.82625 \times 10^1$ ,  $0.82625 \times 10^0$ ,  $0.82625 \times 10^{-1}$

$$N = M \times 10^E$$

- 浮点格式：阶码 + 尾码

- $(-8.2625)_{10}$  转换成浮点形式  $(-0.82625 \times 10^1)_{10}$
    - 将阶码  $(+1)_{10}$  转换成二进制  $(+1)_2$
    - 将尾码  $(-0.82625)_{10}$  转换成二进制  $(-0.110\ 1001\ 1100)_2$

0	001	1	001 0110 0100
阶码 符号位	阶码	尾码 符号位	尾码

- 存储阶码和尾码的二进制编码。注：不同计算机的存储格式可能不同



中國農業大學

閻道宏

# 2.1 数据类型

- 数据类型
  - 计算机存储二进制数据要考虑两个因素，即**存储位数**和**存储格式**
  - 存储非负整数可以使用**无符号格式**；如需存储负数则必须使用**有符号格式**
  - 如需存储实数，则必须使用**浮点格式**，即“阶码+尾码”的存储格式
  - 计算机使用**定长存储**，如果程序员选择不当，则保存数据时可能会出现**溢出或损失精度**等问题
  - 为了让程序员在**申请内存**时能方便地指定存储位数和存储格式，计算机高级语言引入了**数据类型**（data type）的概念
  - Java语言将预定义的数据类型称为**基本数据类型**（primitive data type）



# 2.1 数据类型

- Java语言预定义的8种基本数据类型

数据类型	说明	存储位数 (字节数)	数值范围	运算
byte	整型	8 (1)	-128 ~ 127, 即: $-2^7 \sim 2^7-1$	算术运算 关系运算
short		16 (2)	-32768 ~ 32767, 即: $-2^{15} \sim 2^{15}-1$	
int		32 (4)	$-2^{31} \sim 2^{31}-1$	
long		64 (8)	$-2^{63} \sim 2^{63}-1$	
float	浮点型 (实数)	32 (4)	$3.403 \times 10^{-38} \sim 3.403 \times 10^{38}$ (绝对值精度)	
double		64 (8)	$1.798 \times 10^{-308} \sim 1.798 \times 10^{308}$ (绝对值精度)	
char	字符型	16 (2)	Unicode编码 (UTF-16)	与整数加减 关系运算
boolean	布尔型	8 (1)	true 或 false	逻辑运算





# 2.1 数据类型

## • Java语言与C/C++语言比较

**特别说明：**Java语言与C/C++语言的区别（**基本数据类型**）

- Java语言的整数类型都是有符号格式（signed），没有无符号格式（unsigned）的整数类型。**注：**C/C++语言的整数类型既有有符号格式，也有无符号格式。
- Java数据类型的存储位数是固定的，与操作系统或编译系统无关，其目的是为了跨平台运行。**注：**C/C++语言数据类型的存储位数与操作系统或编译系统有关。
- Java语言的单字节整型为byte。**注：**C/C++语言的单字节整型为char，与字符型相同。
- Java语言的长整型long占8个字节（64位），是int型的两倍。**注：**C/C++语言中，长整型long与int型占用的字节数一样。
- Java语言中的字符型char占2个字节，保存字符的Unicode编码（UTF-16）。**注：**C/C++语言中的字符型char占1个字节，保存字符的ANSI编码。
- Java语言中布尔型的关键字是boolean。**注：**C/C++语言中布尔型的关键字是bool。
- Java语言没有指针类型。例如，下列C/C++用法在Java语言中是错误的。  
    int x, \*p = &x; // C/C++用法：定义一个指向变量x的int型指针变量p  
    \*p = 10; // C/C++用法：通过指针变量p间接访问变量x



中國農業大學

閻道宏

## 2.2 变量与常量

- 温度换算公式： $f = c \times 1.8 + 32$

- 变量

程序中的数据包括原始数据、中间结果、最终结果等，Java语言使用**变量**（variable）来保存这些数据。

- 定义变量：为变量申请内存空间
- 访问变量：写入数据或读出数据
- 定义变量时要考虑三方面的内容
  - 变量如何在内存中存储（存储位数和存储格式）？
  - 变量如何命名？
  - 如何编写定义变量语句？



## 2.2 变量与常量

- 为变量选择数据类型
  - 月份数据：1~12，byte、short、int或long？
  - 温度数据：实数，float或double？
- 为变量命名
  - Java词法元素：关键字、标识符、常量、运算符、分隔符

<i>abstract</i>	<i>assert</i>	<i>boolean</i>	<i>break</i>	<i>byte</i>
<i>case</i>	<i>catch</i>	<i>char</i>	<i>class</i>	<i>const</i>
<i>continue</i>	<i>default</i>	<i>do</i>	<i>double</i>	<i>else</i>
<i>enum</i>	<i>extends</i>	<i>final</i>	<i>finally</i>	<i>float</i>
<i>for</i>	<i>goto</i>	<i>if</i>	<i>implements</i>	<i>import</i>
<i>instanceof</i>	<i>int</i>	<i>interface</i>	<i>long</i>	<i>native</i>
<i>new</i>	<i>null</i>	<i>package</i>	<i>private</i>	<i>protected</i>
<i>public</i>	<i>return</i>	<i>strictfp</i>	<i>short</i>	<i>static</i>
<i>super</i>	<i>switch</i>	<i>synchronized</i>	<i>this</i>	<i>throw</i>
<i>throws</i>	<i>transient</i>	<i>try</i>	<i>void</i>	<i>volatile</i>
<i>while</i>				



## 2.2 变量与常量

- 为变量命名
  - 由程序员定义的程序实体名称被统称为**标识符**（**identifier**），例如变量名
  - 标识符的命名规则
    - 以大写或小写英文字母、下画线 “\_”、美元符号 “\$” 开头
    - 由大写或小写英文字母、下画线 “\_”、美元符号 “\$”、数字 0~9 组成
    - 不能是关键字
  - 标识符举例
    - abc、Abc、\_bc、abc123、abc\_123、A、a、\$No1 ✓
    - 123、abc.123、温度、float ✗



## 2.2 变量与常量

- 定义变量语句的语法

Java语法：变量定义语句

**数据类型** 变量名1, 变量名2, ..., 变量名n ;

语法说明：

- **数据类型**指定了变量的存储位数和存储格式。
- **变量名**需符合标识符的命名规则。
- 可在一条语句中定义多个具有相同数据类型的变量，变量之间用“,” 隔开。

举例：定义两个变量ctemp和ftemp

```
double ctemp ; // 计算机为double型变量ctemp分配8个连续的字节作为其内存单元，  
                // 并将以浮点格式在该内存单元中存储数据
```

```
double ftemp ;
```

或

```
double ctemp, ftemp ; // 在一条语句中定义两个double型的变量
```



中國農業大學

阚道宏

## 2.2 变量与常量

- 访问变量的内存单元
  - 写入（write）数据
    - 从键盘输入数据。例如，  
`Scanner sc = new Scanner( System.in ); // 创建键盘扫描器`  
`ctemp = sc.nextDouble(); // 从键盘接收数据并写入变量ctemp的内存单元`
    - 使用赋值运算符“=”（即等号），对变量进行赋值运算。例如，  
`ctemp = 30 + 6;`
    - 定义时初始化。定义变量的同时为变量赋初始值，这就是初始化。例如，  
`int x=10, y; // y的初始值为null`
  - 读出（read）数据
    - 当变量作为操作数参与运算时，计算机将自动读取其内存单元中存放的数据。例如，  
`ftemp = ctemp*1.8 + 32;`
    - 使用输出语句读出并显示变量内存单元中存放的数据，以使用户查看。例如，  
`System.out.println( ftemp );`
  - 定义后的变量才有内存单元，才能被访问
  - 程序员在编写Java程序时应遵循“先定义，后访问”的原则，未经定义的变量不能访问
  - 不能读取数值为null的变量，否则属于语法错误



## 2.2 变量与常量

- 温度换算公式： $f = c \times 1.8 + 32$
- **变量** (variable)
  - 在编写程序时不能确定其数值大小，例如摄氏温度 $c$ 是今后程序执行时由用户输入的
  - 程序员需要在编写程序时，使用定义变量语句预先为变量分配好内存单元。例如为摄氏温度定义一个变量 $c_{temp}$ ，这样程序执行时才能有内存单元，并能在其中存放摄氏温度数据
- **常量** (constant)
  - 在编写程序时就能确定其数值大小，例如1.8和32
  - 程序员可以将数值直接书写在程序代码中，它们也被称为**字面常量** (literal constant)
  - 不同数据类型常量有不同的**书写形式**



## 2.2 变量与常量

- 常量
  - 整数常量
    - 十进制: 20、-20
    - 八进制: **0**20、-**0**20
    - 十六进制: **0x**20、-**0X**20
    - 二进制: **0b**10100、-**0B**10100
  - 整数常量默认为int型
  - 可以添加后缀“L”（大小写都可以）将其转为long型，例如: 20**L**、-20**L**





## 2.2 变量与常量

- 常量
  - 实数常量（浮点常量）
    - 带小数点：20.5、-20.0。注：-20是整数，而-20.0则是实数
    - 科学记数法：2.05E1或 0.205E2、-2.0E1或 -0.2E2，注：“E”大小写都可以
  - 实数常量默认为double型
  - 可以添加后缀“F”（大小写都可以）将其转为float型，例如：20.5f、2.05e1F



## 2.2 变量与常量

- 常量
  - 字符常量
    - 可见字符: 'A'、'a'、'1'、'中'
    - 转义字符: '\uxxxx'。注: “u”表示Unicode编码, “xxxx”是字符的码值(十六进制)
    - Java预定义转义字符: '\n'、'\t'、'\b'、'\f'、'\r'、'\\"、'\\"、'\\'等
  - Java语言保存一个字符需要占用两个字节, 所保存的是该字符的Unicode编码 (UTF-16)
  - 和英文字符一样, 一个汉字字符也算是一个字符
  - 注: 在C/C++语言中, 一个英文字符占一个字节; 一个汉字字符占两个字节, 算是两个字符。英文字符保存的是ASCII编码, 汉字字符保存的是GBK编码。这种中英文混合编码方式被称为ANSI编码。



## 2.2 变量与常量

- 常量

- 字符串常量

- 可见字符的字符串: "Abc"、"中国China"、"A"、""
    - 带转义字符的字符串:
      - "中国\nChina" 注: 包含一个换行
      - "\"中国\""、"\'China\'" 注: 包含双引号、单引号
      - "C:\\Example\\test.java" 注: 包含反斜杠

- 布尔常量

- 布尔常量只有两个: **true** (真)、**false** (假)



## 2.2 变量与常量

- 只读变量

如果程序所处理的某个数据是常量，在程序运行过程中不需要变动，则可以定义一个**只读变量**（read-only）来保存该数据。

- 只读变量从本质上讲是一个变量，从功能上看就是用变量实现了常量的功能
- 只读变量有时也被简单称作常量
- 和字面常量相比，只读变量可以提高程序可读性、便于调整常量值等优点



## 2.2 变量与常量

### • 只读变量的语法

Java语法：定义只读变量

**final** 数据类型 常变量名 = 初始值 ;

语法说明：

- 使用关键字**final**定义只读变量。
- 只读变量只能被赋值一次。只读变量在取得初始值之后，只能进行**读取**操作，不能做**写入**操作（例如再次赋值）。
- 定义只读变量时通常都会**初始化**。

举例：

```
final int x = 5; // 定义只读变量x，初始值设定为5
x = 10;        // 语法错误：不能对只读变量x再次赋值
final int y;   // 定义只读变量y时没有初始化，此时其数值为null
y = 5;        // 正确：第一次为只读变量y赋值
y = 5;        // 语法错误：不能对只读变量y再次赋值，即使是赋同样的值
```



中國農業大學

阚道宏

## 2.2 变量与常量

- Java语言与C/C++语言比较

特别说明：Java语言与C/C++语言的区别（变量与常量）

- Java变量名可包含美元符号\$。注：C/C++语言不可以。
- 未初始化的Java变量是null，不能读取。注：C/C++语言可以，但读取的是随机值。
- Java语言可以书写二进制整数常量。注：C/C++语言不可以。
- Java语言以Unicode编码（UTF-16）存储字符，一个汉字也是一个字符。注：C/C++语言以ANSI编码存储字符，一个汉字相当于是两个字符。
- Java语言没有“符号常量”，但可通过“只读变量”实现对应的功能。注：C/C++语言可以使用“#define”宏定义指令定义符号常量。




## 2.3 运算符与表达式

- 表达式
  - 运算符、操作数、括号
  - 表达式语句，用于处理数据
- 运算符
  - 优先级、结合性
  - 双目运算符、单目运算符
  - 算术运算、位运算、关系运算、逻辑运算



## 2.3 运算符与表达式

表 2-4 Java 运算符及其优先级、结合性

高优先级	[] . () 注：下标、成员、调用	从左到右
	++ -- + - ~ ! 注：自增、自减、正负号、位反、非	从右到左
	* / %	从左到右
	+ -	从左到右
	<< >> >>>	从左到右
	< > <= >= instanceof	从左到右
	== !=	从左到右
	&	从左到右
	^	从左到右
		从左到右
	&&	从左到右
		从左到右
	? :	从右到左
低优先级	= += -= *= /= %= &= ^=  = <<= >>= >>>=	从右到左





## 2.3 运算符与表达式

- 操作数及其数据类型转换
  - 强制转换

Java语法：数据类型强制转换

(数据类型) 操作数 或 (数据类型) (操作数)

举例：

(short)32指定32为短整型(2字节)

(float)1.8指定1.8为单精度浮点型(4字节)

(long)(-32)指定-32为长整型(8字节)

(double)1.8指定1.8为双精度浮点型(8字节)

注：数据类型应与操作数的数值相符，否则将造成数值的改变。例如，

(float)32将32变为32.0(可以接受)

(byte)129将129变为-127(溢出)

(int)1.8将1.8变为1(丢失小数部分)

(short)32769将32769变为-32767(溢出)

5.5 + 3: 5.5 + (double)3、(int)5.5 + 3

- 自动转换

byte

short

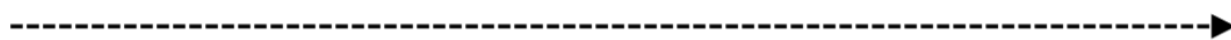
int

long

float

double

低



高



中國農業大學

閻道宏

## 2.3 运算符与表达式

- 表达式结果
  - $5 + 3$ : 8, int
  - $5.5 + 3$ : 8.5, double
  - $5 / 2$ : 2 (不是2.5), int
- 括号
  - Java表达式只使用小括号 “( )”
  - 中括号 “[ ]” 和大括号 “{ }” 被用在其他场合



## 2.3 运算符与表达式

- 其他算术运算符
    - 取正/取负运算符 $+$ 、 $-$ :  $+32$ 、 $-32$ 、 $-x$
    - 取余运算符 $%$ :  $10 \% 6 = 4$
    - 自增运算符 $++$ :  $x++$ 
      - 后置
      - 前置:  $++x$
- `int x = 10; (x++) * 2、(++x) * 2`  
`11      10   20、 11   22`
- 自减运算符 $--$ :  $x--$ 、 $--x$



## 2.3 运算符与表达式

- 位运算
  - 状态位：例如用1表示开，0表示关
  - 位运算符
    - 位反运算符 $\sim$ ：将1变成0，0变成1

$\sim$     0101 0101

=    1010 1010

byte s = 0x55; // (0101 0101)<sub>2</sub>

s =  $\sim$ s;



中國農業大學

閻道宏

## 2.3 运算符与表达式

- 位运算

- 位与运算符**&**：参与运算的两个位都为1，则结果为1，否则为0

bbbb bbbb	操作数 s，其中 b 表示 0 或 1
& 0000 0010	检测倒数第 2 位状态的掩码 (0x02)
= 0000 00b0	运算结果：保留倒数第 2 位，其他位变成 0
	如果位与结果为 0（即 8 位全部为 0），则倒数第 2 位的状态为 0； 否则倒数第 2 位的状态为 1

- s & 0x02

bbbb bbbb	操作数 s，其中 b 表示 0 或 1
& 1111 1101	将倒数第 2 位状态置 0 的掩码 (0xFD)
= bbbb bb0b	运算结果：将倒数第 2 位置成 0，其他位不变

- s = s & 0xFD;



## 2.3 运算符与表达式

- 位运算
  - 位或运算符 **|**：参与运算的两个位只要有一位为1，则结果为1，否则为0

$$\begin{array}{r} 0011\ 0011 \\ | \quad 0000\ 1111 \\ = \quad 0011\ 1111 \end{array}$$

bbbb bbbb	操作数 s，其中 b 表示 0 或 1
0000 0010	将倒数第 2 位状态置 1 的掩码 (0x02)
= bbbb bb1b	运算结果：将倒数第 2 位置成 1，其他位不变

- $s = s | 0x02;$



## 2.3 运算符与表达式

- 位运算

- 异或运算符<sup>^</sup>: 参与运算的两个位不同（0和1，或1和0），则结果为1，否则为0

bbbb bb0b	操作数 s，其中 b 表示 0 或 1。假设倒数第 2 位为 0
<sup>^</sup> 0000 0010	将倒数第 2 位状态进行反置的掩码（0x02）
= bbbb bb1b	运算结果：将倒数第 2 位由 0 反置成 1，其他位不变
bbbb bb1b	操作数 s，其中 b 表示 0 或 1。假设倒数第 2 位为 1
<sup>^</sup> 0000 0010	将倒数第 2 位状态进行反置的掩码（0x02）
= bbbb bb0b	运算结果：将倒数第 2 位由 1 反置成 0，其他位不变

- $s = s \wedge 0x02;$



## 2.3 运算符与表达式

- 位运算

- 左移运算符<<: 将操作数按二进制位左移指定的位数，左移时高位被移除，低位补0

	0011 0011	8 位操作数
<<	2	左移 2 位
=	<del>00</del> 1100 11 <u>00</u>	高 2 位被移除，低 2 位补 0，得到 1100 1100

- 操作数 << 左移位数
- $s << 2$





## 2.3 运算符与表达式

- 位运算

- 右移运算符：将操作数按二进制位右移指定的位数，右移时低位被移除。高位怎么补呢？
  - 带符号右移 $\gg$ ：高位补符号位
  - 不带符号右移 $\ggg$ ：高位补0

	1 011 0011	8 位操作数，最高位为符号位（1 表示负数）
$\gg$	2	带符号右移 2 位
=	<u>1</u> 110 1100 <del>11</del>	低 2 位被移除，高 2 位补符号位（1），得到 1110 1100

	1 011 0011	8 位操作数，最高位为符号位（1 表示负数）
$\ggg$	2	不带符号右移 2 位
=	<u>0</u> 010 1100 <del>11</del>	低 2 位被移除，高 2 位补 0，得到 0010 1100

- 操作数  $\gg$  右移位数、操作数  $\ggg$  右移位数
- $s \gg 2$ 、 $s \ggg 2$



## 2.3 运算符与表达式

- 赋值运算符=

用于修改变量的数值，即将新数值写入变量对应的内存单元，存储在该内存单元中的原数值将被覆盖。

```
int x = 0, y = 0;
```

```
x = 5;
```

```
y = x + 3;
```

— 赋值运算本身也构成一个赋值表达式

```
x = 5、(x = 5) * 2
```

```
y = x = 2 + 6、y = ( x = (2+6) )
```



中國農業大學

阚道宏

## 2.3 运算符与表达式

- 泛化的运算符：=、++、--
  - 合理运用泛化运算符可以让语句更加简洁

语句：a = 10; b = 10; c = 10; 可简写成：a = b = c = 10;

语句：y = x; x = x + 1; 可简写成：y = x++;

语句：x = x + 1; y = x; 可简写成：y = ++x;

语句：y = x; x = x - 1; 可简写成：y = x--;

语句：x = x - 1; y = x; 可简写成：y = --x;



## 2.3 运算符与表达式

- 复合赋值运算符

表2-5 复合赋值运算符（共11个）

+=	-=	*=	/=	%=	&=	=	^=	<<=	>>=	>>>=
----	----	----	----	----	----	---	----	-----	-----	------

—  $x \text{ ?} = \text{exp}$  等价于  $x = x \text{ ?} ( \text{exp} )$

$x = x + 5;$  可简写成:  $x += 5;$

特别说明: Java语言与C/C++语言的区别（运算符与表达式）

- Java语言没有无符号数的概念，都是有符号数。有符号数在右移时分带符号右移（>>）和不带符号右移（>>>）两种。



## 2.4 算法结构与控制语句

- 三种算法基本结构
  - 顺序结构、选择结构、循环结构
  - 条件：真（true）、假（false）
  - 布尔类型（boolean）
    - 关系运算符：例如，大于、小于、等于
    - 逻辑运算符：与、或、非
- Java语言：选择语句和循环语句



## 2.4 算法结构与控制语句

- 布尔类型及其运算
  - 布尔类型：boolean、true、false
  - 关系运算符：用于比较两个数之间的大小

表2-6 关系运算符（共6个）

>（大于）	>=（大于等于）	<（小于）	<=（小于等于）	==（等于）	!=（不等于）
-------	----------	-------	----------	--------	---------

例2-1 关系表达式举例

关系表达式	布尔型结果	备注
5 > 3	true	5大于3吗？是的
5 >= 3	true	5大于或等于3吗？是的
5 <= 3	false	5小于或等于3吗？不是
5 == 3	false	5等于3吗？不是
5 != 3	true	5不等于3吗？是的
2+3 <= 1+2	false	比较两个算术表达式时，先计算表达式，再比较其结果。 算术运算符优先级高于关系运算符。



## 2.4 算法结构与控制语句

- 布尔类型及其运算
  - 关系运算符

例2-2 由关系表达式所描述的条件举例（假设：int x = 10;）

条件	布尔型结果	条件是否成立
$x > 5$	true	x大于5吗？是的，条件成立
$x < 5$	false	x小于5吗？不是，条件不成立
$x - 5 == 5$	true	x - 5等于5吗？是的，条件成立
$x - 5 < 0$	false	x - 5小于0吗？不是，条件不成立



## 2.4 算法结构与控制语句

- 布尔类型及其运算
  - 逻辑运算符

表2-7 逻辑运算符（共3个）

逻辑运算符	运算规则
<b>&amp;&amp;</b> （逻辑与）	双目运算符。若两个操作数都为true，则结果为true；否则为false。相当于“并且”的意思。
<b>  </b> （逻辑或）	双目运算符。若两个操作数中有一个为true，则结果为true；否则为false。相当于“或”的意思。
<b>!</b> （逻辑非）	单目运算符。若操作数为true则结果为false；若操作数为false则结果为true。相当于“求反”的意思。

例2-3 由逻辑表达式所描述的复合条件举例（假设：int x=10, y=20;）

复合条件	布尔型结果	条件是否成立
x > 5 && y > 10	true	条件成立
x < 5    y < 10	false	条件不成立
x - 5 == 5    y == 0	true	条件成立
!(x > 5)	false	条件不成立





## 2.4 算法结构与控制语句

- 选择语句
  - 有些算法，其中的某些操作步骤需满足特定条件才被执行

例2-4 算法举例：给定x的值，求其倒数

- 1 定义变量x，申请保存数值的内存空间。
- 2 从键盘输入变量x的值。
- 3 如果条件“x不等于0”成立，则转到4计算倒数，否则转到5提示错误信息。
- 4 计算并显示表达式 $1/x$ 的结果，转6。
- 5 条件“x不等于0”不成立（即x等于0），显示错误信息。
- 6 算法结束

- 如果.....，就.....，否则.....
- 如果**条件**成立，则执行**算法分支1**，否则执行**算法分支2**
- 选择语句：if-else、switch-case



## 2.4 算法结构与控制语句

### • 选择语句

Java语法: **if-else**语句

```
if (表达式)
{ 语句1 }
else
{ 语句2 }
```

语法说明:

- **表达式**指定一个判断条件。该表达式结果应为布尔类型，例如关系表达式或逻辑表达式。
- **语句1**是描述算法分支1的Java语句序列，即条件成立时执行的语句序列。
- **语句2**是描述算法分支2的Java语句序列，即条件不成立时执行的语句序列。如果条件不成立时不需要做什么处理，则省略**else**和{ 语句2 }。
- 语句1、语句2可能是包含多条Java语句的序列，此时必须用一对大括号{ }将它们括起来。如果只包含一条语句，则大括号可以省略。
- 计算机执行该语句时，首先计算表达式（即判断条件），若结果为**true**（条件成立），则执行语句1；否则，执行**else**后面的语句2。



中國農業大學

阚道宏

## 2.4 算法结构与控制语句

- 选择语句

例2-5 实现求倒数算法的Java程序（if-else语句）

```
1 import java.util.Scanner; // 导入外部程序Scanner
2
3 public class JavaTest { // 主类
4     public static void main(String[] args) { // 主方法
5         Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6         double x; // 定义一个double型变量x
7         x = sc.nextDouble(); // 键盘输入变量x的值
8
9         if (x != 0) { // 判断条件“x不等于0”是否成立
10             // 条件成立时执行下列代码。因为是多条语句，所以用{}括起来
11             double y; // 再定义一个double型变量y，用于保存x的倒数
12             y = 1 / x; // 求x的倒数，结果赋值给y
13             System.out.println( y ); // 显示y的值，即x的倒数
14         }
15         else
16             System.out.println( "0的倒数没有意义" ); // 显示错误信息
17             // else分支只有一条语句，可省略大括号
18     }
19 }
```

复合语句  
空语句



## 2.4 算法结构与控制语句

- 选择语句

例2-6 判断年份是否闰年的Java程序

```
1 import java.util.Scanner; // 导入外部程序Scanner
2
3 public class JavaTest { // 主类
4     public static void main(String[] args) { // 主方法
5         Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6         int year;      // 定义一个int型变量year
7         year = sc.nextInt(); // 键盘输入一个年份，保存到变量year中
8
9         if ( (year%4 == 0 && year%100 != 0) || year%400 == 0 ) // 判断闰年条件是否成立
10            System.out.println( year + "是闰年" );      // 条件成立则该年份是闰年
11        else
12            System.out.println( year + "不是闰年" );      // 否则该年份不是闰年
13    }
14 }
```



## 2.4 算法结构与控制语句

- 选择语句

例2-7 求符号函数sgn(x)的Java程序

```
1 import java.util.Scanner; // 导入外部程序Scanner
2
3 public class JavaTest { // 主类
4     public static void main(String[] args) { // 主方法
5         Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6         float x; // 定义一个float型变量x
7         x = sc.nextFloat(); // 键盘输入变量x的值
8
9         int sgn; // 定义一个int型变量sgn, 用于保存符号函数的结果
10        if (x == 0) // 首先将x分为等于0和不同于0两种情况
11            sgn = 0; // x = 0的情况
12        else { // 在x不等于0时, 再进一步区分x>0和x<0这两种情况
13            if (x > 0) sgn = 1; // x > 0的情况
14            else sgn = -1; // x < 0的情况
15        }
16        System.out.println( sgn ); // 显示sgn的值, 即符号函数的结果
17    }
18 }
```

$$\text{sgn}(x) = \begin{cases} 1 & (x > 0) \\ 0 & (x = 0) \\ -1 & (x < 0) \end{cases}$$



## 2.4 算法结构与控制语句

- 选择语句
  - if - else if

例2-8 求符号函数sgn(x)的Java程序（if-else if语句）

```
1 import java.util.Scanner; // 导入外部程序Scanner
2
3 public class JavaTest { // 主类
4     public static void main(String[] args) { // 主方法
5         Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6         float x; // 定义一个float型变量x
7         x = sc.nextFloat(); // 键盘输入变量x的值
8
9         int sgn; // 定义一个int型变量sgn，用于保存符号函数的结果
10        if (x == 0) sgn = 0; // 首先检查x等于0的情况
11        else if (x > 0) sgn = 1; // 再检查x大于0的情况
12        else sgn = -1; // 最后剩下的就是x小于0的情况
13        System.out.println( sgn ); // 显示符号函数的结果
14    }
15 }
```



中國農業大學

阚道宏

## 2.4 算法结构与控制语句

### • 选择语句

Java语法: if-else if语句

```
if (表达式1) 语句1  
else if (表达式2) 语句2  
.....  
else if (表达式n) 语句n  
else 语句n+1
```

语法说明:

- 表达式1~n分别是需依次判断的条件。表达式结果应为布尔类型，例如关系表达式或逻辑表达式。
- 语句1~n分别对应条件成立时执行的语句，可以是单条语句、复合语句或空语句。
- 语句n+1是所有条件都不成立时执行的语句，可以是单条语句、复合语句。如果所有条件都不成立时不需要做什么处理，即空语句，则省略else和语句n+1。
- 计算机执行该语句时，首先计算表达式1，若为true则执行语句1；否则继续计算表达式2，.....，直到表达式n；如果所有条件都不成立则执行else后面的语句n+1。计算机只会执行语句1~n+1中的一条。



中國農業大學

阚道宏

## 2.4 算法结构与控制语句

- 选择语句

例2-9 显示星期几英文单词的Java程序

```
1 import java.util.Scanner; // 导入外部程序Scanner
2
3 public class JavaTest { // 主类
4     public static void main(String[] args) { // 主方法
5         Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6         int x; // 定义一个int型变量x
7         x = sc.nextInt(); // 键盘输入一个表示星期几的数值（1~7），保存到变量x中
8
9         // 下列if-else if语句根据x的值显示其对应的英文单词
10        if (x == 1) System.out.println( "Monday" );
11        else if (x == 2) System.out.println( "Tuesday" );
12        else if (x == 3) System.out.println( "Wednesday" );
13        else if (x == 4) System.out.println( "Thursday" );
14        else if (x == 5) System.out.println( "Friday" );
15        else if (x == 6) System.out.println( "Saturday" );
16        else if (x == 7) System.out.println( "Sunday" );
17        else System.out.println( "Input Error" ); // 输入数值不在1~7范围之内，提示错误
18    }
19 }
```





## 2.4 算法结构与控制语句

- 选择语句
  - 条件运算符 “?:”  
int a = 5, b = 10, c;  
if (a > b) c = a;  
else c = b;

Java语法：条件运算符 “?:”

表达式 ? 表达式1 : 表达式2

语法说明：

- 条件运算符将3个表达式连接在一起，构成一个大的**条件表达式**。其中的**表达式**指定一个判断条件，该表达式结果应为布尔类型，例如关系表达式或逻辑表达式。
- 如果表达式的结果为true，则计算**表达式1**，将其结果作为整个条件表达式的结果；否则计算**表达式2**，将其结果作为整个条件表达式的结果。
- 条件运算符为3目运算符。

举例：int a = 5, b = 10, c;

```
a > b ? a : b // 这是一个条件表达式，其结果等于10，数据类型为int型
System.out.println( a > b ? a : b ); // 显示条件表达式的结果
c = ( a > b ? a : b ); // 将条件表达式的结果赋值给变量c
```



## 2.4 算法结构与控制语句

- 选择语句

- switch-case语句：多分支结构算法

例2-10 一类特殊的多分支结构算法

- 1 计算某个表达式，判断其结果属于下列哪种情况。
- 2 情况1：执行算法分支1，执行结束转到7。
- 3 情况2：执行算法分支2，执行结束转到7。
- 4 .....
- 5 情况n：执行算法分支n，执行结束转到7。
- 6 否则属于其他情况：执行算法分支n+1，执行结束转到7。
- 7 算法结束。



## 2.4 算法结构与控制语句

Java语法：switch-case语句

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

语法说明：

- 计算机执行该语句时，首先计算switch后面的表达式，然后将结果依次与各case后的常量表达式的结果进行比对。若比对成功，则以比对成功的case语句为起点，顺序执行后面的所有语句，直到整个switch-case语句结束；或遇到break语句时中途跳出switch-case语句。如果所有比对都不成功，则将default语句作为执行的起点。
- 表达式的结果应当是整型或字符型（即byte、short、int、long或char型），不能是浮点型。
- 常量表达式1~n分别列出switch后面“表达式”可能的结果。常量表达式只能是常量，或由常量组成的表达式。各常量表达式的结果不能相同。
- 语句1~n分别对应常量表达式比对成功时应执行的语句序列。通常都在末尾增加一条break语句，这样可以宣告算法结束，中途跳出。
- 语句n+1是default后面的语句，即所有比对都不成功时应执行的语句。default语句习惯上被放在最后。语句1~n+1为复合语句时，大括号也可省略。



中國農業大學

阚道宏

## 2.4 算法结构与控制语句

例2-11 显示星期几英文单词的Java程序（switch-case语句）

```
1 import java.util.Scanner; // 导入外部程序Scanner
2
3 public class JavaTest { // 主类
4     public static void main(String[] args) { // 主方法
5         Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6         int x; // 定义一个int型变量x
7         x = sc.nextInt(); // 键盘输入一个表示星期几的数值（1~7），保存到变量x中
8         // 下列switch-case语句根据x的值显示对应的英文单词
9         switch ( x ) {
10             case 1: System.out.println( "Monday" ); break;
11             case 2: System.out.println( "Tuesday" ); break;
12             case 3: System.out.println( "Wednesday" ); break;
13             case 4: System.out.println( "Thursday" ); break;
14             case 5: System.out.println( "Friday" ); break;
15             case 6: System.out.println( "Saturday" ); break;
16             case 7: System.out.println( "Sunday" ); break;
17             default: System.out.println( "Input Error" ); break;
18         }
19         // 每个case语句显示出对应的英文单词之后，程序功能即已完成
20         // 因此使用break语句中途跳出switch语句
21     } }
```



## 2.4 算法结构与控制语句

例2-12 显示不同月份天数的Java程序（switch-case语句：共用语句）

```
1 import java.util.Scanner; // 导入外部程序Scanner
2
3 public class JavaTest { // 主类
4     public static void main(String[] args) { // 主方法
5         Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6         int month; // 定义一个int型变量month
7         month = sc.nextInt(); // 键盘输入一个月份（1~12），保存到变量month中
8         // 下列switch-case语句显示不同月份的天数
9         switch ( month ) {
10            case 1: // 1月大
11            case 3: // 3月大
12            case 5: // 5月大
13            case 7: // 7月大
14            case 8: // 8月大
15            case 10: // 10月大
16            case 12: System.out.println( "31天" ); break; // 1、3、5、7、8、10、12月共用语句
17            case 4: // 4月小
18            case 6: // 6月小
19            case 9: // 9月小
20            case 11: System.out.println( "30天" ); break; // 4、6、9、11月共用语句
21            case 2: System.out.println( "28或29天" ); break; // 2月
22            default: System.out.println( "Input Error" ); break; // 提示错误信息
23        }
24    } }
```



## 2.4 算法结构与控制语句

- 循环语句
  - 有一些算法，在满足特定条件下将重复执行某些操作步骤
  - 奇数数列：1, 3, 5, 7, 9, ..., 求数列前 $N$ 项的累加和

$$\sum_{n=1}^N 2n - 1$$

- 如果.....，就重复做.....，否则停止
- 如果**条件**成立，则重复执行**循环体**，否则**结束**循环
- 循环的4个要素
  - 循环变量、循环变量的初始值、循环条件、循环体



## 2.4 算法结构与控制语句

- 循环语句

例2-13 算法举例：求解奇数数列前N项累加和的循环结构算法

- 1 首先定义一个int型变量N，从键盘输入N的值。
- 2 定义一个循环变量n（初始值为1），表示当前数列项的序号。
- 3 再定义一个int型变量sum（初始值为0），用于保存累加的结果。
- 4 开始循环：如果循环条件 $n \leq N$ 成立，则转到5做累加操作，否则转到7结束循环。
- 5 将当前项的值 $2n-1$ 累加到sum上： $sum += 2n - 1$ 。
- 6 将n加1，准备下一次累加，转到4继续循环。步骤5~6是被重复执行的循环体。
- 7 循环结束后，显示sum的值，此时sum中的值就是数列前N项的累加和。
- 8 算法结束。

– 3种循环语句：**while**、**do-while**、**for**



中國農業大學

閻道宏

## 2.4 算法结构与控制语句

### • 循环语句

Java语法：while语句

**while** (表达式)  
语句

语法说明：

- **表达式**指定一个循环条件。该表达式结果必须布尔类型，例如关系表达式或逻辑表达式。
- **语句**是描述循环体的Java语句，即条件成立时循环执行的算法。如循环条件一开始就不成立，则循环体一次也不执行。循环体中应包含使循环条件趋向于false的语句，否则循环条件一直为true，循环体将无休止地执行，俗称为**死循环**。
- 计算机执行该语句时，首先计算表达式（即循环条件），若结果为true（条件成立），则重复执行循环体语句；否则结束循环。



中國農業大學

閻道宏



## 2.4 算法结构与控制语句

- 循环语句

例2-14 求解奇数数列前N项累加和的Java程序（while语句）

```
1  import java.util.Scanner; // 导入外部程序Scanner
2
3  public class JavaTest { // 主类
4      public static void main(String[] args) { // 主方法
5          Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6          int N; // 定义一个int型变量N
7          N = sc.nextInt(); // 键盘输入变量N的值
8
9          int n = 1, sum = 0; // 定义循环变量n（初始值为1），
10             // 定义保存累加结果的变量sum（初始值为0）
11          while (n <= N) { // 用小括号将循环条件n<=N括起来
12              sum += 2*n - 1; // 将当前项的值2n-1累加到sum上
13              n++; // 将n加1，准备下一次累加。该语句使得循环条件n<=N趋向于false
14              // 执行完循环体最后一条语句之后，转到第11行，重新判断循环条件
15          }
16          // 如果循环条件不成立，则循环结束，继续执行while语句的下一条语句
17          System.out.println( sum ); // 显示变量sum的值，即前N项的累加和
18      }
19  }
```



## 2.4 算法结构与控制语句

### • 循环语句

Java语法：do-while语句

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

语法说明：

- **表达式**指定一个循环条件。将条件放在循环体语句的后面，即先执行，再判断条件。该表达式结果必须布尔类型，例如关系表达式或逻辑表达式。
- **语句**是描述循环体的Java语句，不管循环条件是否成立，循环体至少执行一次。如果循环体只包含一条语句，则大括号“{ }”可以省略。循环体中应包含使循环条件趋向于false的语句，否则将造成死循环。
- 计算机执行该语句时，首先执行一次循环体，然后再计算表达式（即循环条件），若结果为true（条件成立），则重复执行循环体语句；否则结束循环。



中國農業大學

阚道宏

## 2.4 算法结构与控制语句

- 循环语句

例2-15 求解奇数数列前N项累加和的Java程序（do-while语句）

```
1 import java.util.Scanner; // 导入外部程序Scanner
2
3 public class JavaTest { // 主类
4     public static void main(String[] args) { // 主方法
5         Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6         int N; // 定义一个int型变量N
7         N = sc.nextInt(); // 键盘输入变量N的值
8
9         int n = 1, sum = 0; // 定义循环变量n（初始值为1），
10            // 定义保存累加结果的变量sum（初始值为0）
11         do { // 先执行循环体
12             sum += 2*n - 1; // 将当前项的值2n-1累加到sum上
13             n++; // 将n加1，准备下一次累加
14         } while (n <= N); // 后判断条件。如条件成立则重复执行循环体，否则结束循环
15         // 循环结束后，继续执行do-while语句的下一条语句
16         System.out.println( sum ); // 显示变量sum的值，即前N项的累加和
17     }
18 }
```



中國農業大學

閻道宏

## 2.4 算法结构与控制语句

### • 循环语句

Java语法: for语句

**for** (表达式1; 表达式2; 表达式3)  
语句

语法说明:

- **表达式1**只在正式循环前执行一次，通常用于为循环算法赋初始值。
- **表达式2**指定一个循环条件。每次循环时，先计算该表达式，如果为**true**则执行下面的循环体语句，否则结束循环。
- **表达式3**在每次循环体执行结束之后都被执行一次，主要用于修改循环条件中的某些变量，使循环条件趋向于**false**。
- **语句**是描述循环体的Java语句。
- 计算机执行该语句时，首先计算表达式1（通常为赋初始值）；再计算表达式2（即循环条件），若结果为**true**则重复执行循环体语句，每次执行完循环体语句之后都计算一次表达式3（通常用于修改循环条件中的某些变量），然后再返回表达式2重新判断条件；若表达式2的结果为**false**则结束循环。



中國農業大學

閻道宏

## 2.4 算法结构与控制语句

- 循环语句

例2-16 求解奇数数列前N项累加和的Java程序（for语句）

```
1 import java.util.Scanner; // 导入外部程序Scanner
2
3 public class JavaTest { // 主类
4     public static void main(String[] args) { // 主方法
5         Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6         int N; // 定义一个int型变量N
7         N = sc.nextInt(); // 键盘输入变量N的值
8
9         int n, sum = 0; // 定义循环变量n
10        // 定义保存累加结果的变量sum（初始值为0）
11        for (n = 1; n <= N; n++) { // for语句集中用3个表达式指定n的初始值1、循环条件n<=N
12            // 以及修改循环变量n++，使循环条件趋向于false
13            sum += 2*n - 1; // 循环体被简化了，原来的n++语句被放入到for语句里面
14        } // 循环体只有一条语句，此时这对大括号可以省略
15        System.out.println( sum ); // 显示变量sum的值，即前N项的累加和
16    }
17 }
```



## 2.4 算法结构与控制语句

- break语句和continue语句

- 控制语句

```
int a=5, b=10, c; // 将a、b中较大的数赋值给c
```

```
if (a > b)
```

```
    c = a;
```

```
else
```

```
    c = b;
```

```
System.out.println( c );
```

- 造成程序执行顺序跳转的语句被统称为控制语句



## 2.4 算法结构与控制语句

- break语句

例2-18 一个计算圆面积的Java程序（break语句应用示例）

```
1 import java.util.Scanner; // 导入外部程序Scanner
2
3 public class JavaTest { // 主类
4     public static void main(String[] args) { // 主方法
5         Scanner sc = new Scanner( System.in ); // 创建扫描器对象sc
6         double r; // 定义一个变量r来存放圆的半径
7
8         while ( true ) { // 死循环
9             r = sc.nextDouble(); // 键盘输入圆的半径
10            if (r <= 0) break; // 如果用户输入的半径小于或等于0，则跳出循环
11            System.out.println( 3.14*r*r ); // 显示圆面积
12        }
13        // 使用break语句中途跳出while语句，继续执行while语句的下一条语句
14    }
15 }
```

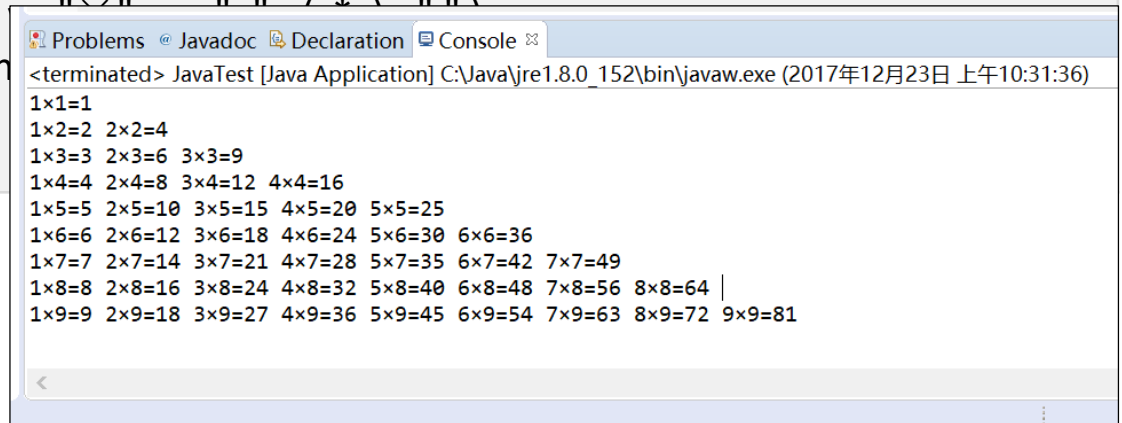


## 2.4 算法结构与控制语句

- 多重循环

例2-19 生成乘法表的Java程序（多重循环应用示例）

```
1 public class JavaTest { // 主类
2
3     public static void main(String[] args) { // 主方法
4         int x, y;           // 定义两个循环变量x和y
5         for (x = 1; x <= 9; x++) { // 第一重循环，x从1到9，共9行
6             for (y = 1; y <= x; y++) // 第二重循环，y从1到x。第x行有x个乘法
7                 System.out.print(" ");
8             System.out.print("\n");
9         }
10    }
```



Problems @ Javadoc Declaration Console

```
<terminated> JavaTest [Java Application] C:\Java\jre1.8.0_152\bin\javaw.exe (2017年12月23日 上午10:31:36)
1x1=1
1x2=2 2x2=4
1x3=3 2x3=6 3x3=9
1x4=4 2x4=8 3x4=12 4x4=16
1x5=5 2x5=10 3x5=15 4x5=20 5x5=25
1x6=6 2x6=12 3x6=18 4x6=24 5x6=30 6x6=36
1x7=7 2x7=14 3x7=21 4x7=28 5x7=35 6x7=42 7x7=49
1x8=8 2x8=16 3x8=24 4x8=32 5x8=40 6x8=48 7x8=56 8x8=64
1x9=9 2x9=18 3x9=27 4x9=36 5x9=45 6x9=54 7x9=63 8x9=72 9x9=81
```



中国农业大学

阚道宏



## 2.4 算法结构与控制语句

- continue语句

例2-20 显示1~50之间所有能被3整除的数（continue语句应用示例）

```
1 public class JavaTest { // 主类
2
3     public static void main(String[] args) { // 主方法
4         for (int n = 1; n <= 50; n++) { // 从1到50的循环
5             if (n%3 != 0) continue; // 如果n不能被3整除，则执行continue语句
6             // continue语句的作用是结束本次循环，中途返回，去检查下一个数
7             // 未中途返回的数是能被3整除的数，下面将显示这些数并用逗号隔开
8             System.out.print( n +", " );
9         }
10    } }
```



## 2.4 算法结构与控制语句

- 带标号的break和continue

**Label1:** for ( ..... ) { // 为外层循环语句添加标号, 假设为Label1

**Label2:** for ( ..... ) { // 为内层循环语句添加标号, 假设为Label2

```
.....  
    break Label1; // 或: continue Label1;  
}  
}
```

- 带标号的break语句可直接跳出标号指定的循环, 多层循环时可以是任意的外层循环
- 带标号的continue语句可直接返回标号指定的循环, 多层循环时可以是任意的外层循环



## 2.4 算法结构与控制语句

- 带标号的break和continue

例2-21 显示100~200之间的所有质数（带标号的continue语句应用示例）

```
1 public class JavaTest { // 主类
2     public static void main(String[] args) { // 主方法
3         int i, j, n = 0;
4
5         Loop1: for (i = 100; i <= 200; i += 2) { // 外层循环，语句块标号Loop1
6             Loop2: for(j = 2; j <= i/2; j++) { // 内层循环，语句块标号Loop2
7                 if (i%j == 0) // 不是质数，则中途返回
```

特别说明：Java语言与C/C++语言的区别（控制语句）

- Java语言选择语句和循环语句里的条件表达式必须是布尔型，不能是其他类型。注：C/C++语言可以自动将其他类型转为布尔型，非0值转成true，0转成false。
- Java语言里的break和continue语句可以带标号，直接跳出外层循环，或直接返回外层循环。注：C/C++语言没有带标号的break和continue语句。



# 第2章 Java语言基础

- 本章学习要点

- Java语言的**基础语法**大量借鉴了C/C++语言。具有C/C++语言基础的读者在学习Java语言时，只需重点了解它与C/C++语言之间的区别
- 本章应尽快熟悉Java语言**编程环境**。建议把之前学习过的C/C++程序改用Java语言重写一遍
- 通过对比可以知道，程序设计语言可以是不同的，但算法设计思想是共同的

