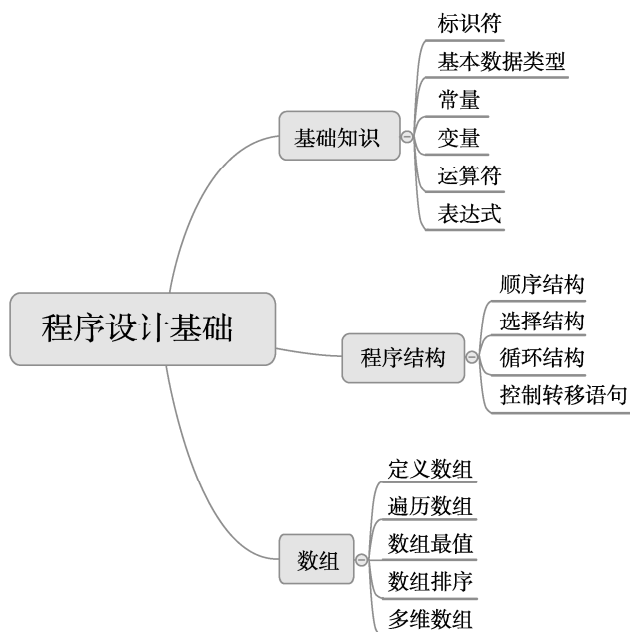


第 2 章 程序设计基础

内容导览



课件



源代码



AI 提示词

学习目标

- ❑ 能够运用程序设计基础知识编写简单 Java 程序；
- ❑ 掌握程序控制结构并能够应用其进行程序问题的分析和设计；
- ❑ 能够运用 `break`、`continue` 和 `return` 语句实现程序执行流程的转移；
- ❑ 掌握数组的定义和遍历等操作，运用数组解决排序、最大值与最小值问题。

2.1 引例

为了说明数据类型、一维数组及其定义，先讨论下面的例题。

【例 2.1】 某专业 50 名学生在上学期学习了程序设计课程。设计一个程序，从键盘输入 50 名学生的考试成绩，计算该课程的平均成绩和及格率，平均成绩要精确到小数点后一位，及格率保留一位小数，以百分数形式表示。

分析：从直观上看，程序处理数据的过程与人工处理的过程基本相似。下面借鉴逐步求精的

方法分析问题的求解过程。根据第1章的讨论，已知Java程序是由类组成的。因此可以将问题转化为“设计一个成绩分析类”。由于暂时还没有学习类与对象的定义方法，先假设已经定义好了相应的类，只需直接考虑数据处理逻辑。

第1步求精：直观分析问题的求解过程。

- 输入50名学生的成绩；
- 计算平均成绩；
- 统计及格率；
- 输出结果。

第2步求精：细化实现方法。

- 定义表示50名学生课程成绩的数组，设计输入该课程成绩的方法；
- 定义表示平均成绩的变量，设计计算平均成绩的公式

平均成绩=(50名学生成绩之和)/50

- 定义一个整型变量，设计统计及格人数的方法，若学生成绩 ≥ 60 ，及格人数加1；
- 定义表示及格率的变量，设计计算及格率的方法，及格率=及格人数 $\div$ 50；
- 设计结果输出的方式。

第3步求精：设计并定义类。

由于Java程序是由类组成的，现在必须考虑定义什么样的类。根据第1章的内容，定义类一般可以分为三个步骤。现在考虑将第2步求精的结果作为类的main()方法。

- ① 引入标准类库；
- ② 定义类名；
- ③ 设计类的main()方法：

- 定义表示课程成绩的数组、及格人数的变量及平均成绩和及格率的变量；
- 利用循环输入50名学生的成绩；
- 在循环体中统计及格人数；
- 计算平均成绩和及格率；
- 输出结果（通常需要确定输出格式，此处暂不追求完美的输出格式）。

将以上描述转换成Java代码，程序源代码如下。

```
//文件名 Jpro2_1.java
1  import java.util.Scanner;
2  public class Jpro2_1 {
3      public static void main(String[] args) {
4          int count_score = 0;                //统计及格人数
5          int sum = 0;                        //累加成绩总和
6          float average_score;
7          float rate_score;
8          int[] pro_score = new int[50];      //存放50名学生的成绩
9          Scanner scanner = new Scanner(System.in);
10         for (int i = 0; i < 50; i++) {
11             pro_score[i] = scanner.nextInt();
12             if (pro_score[i] >= 60)
13                 count_score++;
14             sum = sum + pro_score[i];
15         }
16         average_score = sum / 50.0f;        //计算平均成绩
```

```

17         rate_score = count_score / 50.0f;           //计算及格率
18         System.out.println("The average score is:" + average_score);
19         System.out.println("The rate score is:" + rate_score);
20         scanner.close();
21     }
22 }

```

程序分析：程序第1行导入 `java.util.Scanner` 类，用于从键盘读取数据。第4、5行声明两个整型变量，分别用于统计及格人数和累加成绩总和。第8行定义长度为50的整型数组 `pro_score`，存放学生成绩。第9行创建 `Scanner` 对象。第10~15行为 `for` 循环，循环50次，每次读取一个整数存入数组，并通过 `if` 语句判断是否及格以统计及格人数，同时累加成绩总和。第16行计算平均成绩，除以 `50.0f` 确保结果为浮点数。第17行计算及格率。第18、19行输出结果。第20行关闭 `Scanner` 对象释放资源。

如果没有这些语句，下面的计算公式，例如 `sum=sum+pro_score[i]`（在Java中被称为表达式）将无法正常运行并得出结果。也就是说，在Java程序中，必须先定义变量，再使用。很显然，为了正确地定义变量，必须先理解实际数据的性质，确定其类型。可以将上述第三步求精过程进一步描述如下。

① 引入标准类库；

② 定义类名；

③ 设计类的 `main()` 方法：

- ❑ 确定原始数据的数据类型并设计表示它们的变量名（及格人数用 `count_score` 表示，成绩总和用 `sum` 表示，学生人数用 `i` 表示，指定其为整数，用 `int` 类型表示；用整型数组 `pro_score[]` 存储同学的考试成绩）。
- ❑ 确定程序运行过程中用到的变量及其类型（用 `average_score` 表示平均成绩，用 `rate_score` 表示及格率，考虑到成绩要精确到小数点后面一位，用浮点型表示）。
- ❑ 定义变量（前两步分析在程序中反映不出来，只是为这一步建立基础）。
- ❑ 设计循环输入50名学生成绩存储到数组 `pro_score` 中。
- ❑ 在循环体中计算50分同学成绩之和 `sum`，计算平均成绩 `average_score=sum/50.0`。
- ❑ 根据循环体中统计的及格人数 `count_score`，计算及格率 `rate_score=count_score/50.0`。
- ❑ 输出结果。

实际上，这里的分析过程并不完全符合面向对象的要求，在第3章将有进一步的讨论。另外，计算平均成绩的公式与通常的数学公式的表示方式并不相同，这也是程序设计语言的特定要求。详细的情况将在本章后面的表达式一节中讨论。本例中50名学生的成绩数据使用了数组来表示，至于什么是数组，为什么要用数组，将在2.4节中解释。本例中涉及的变量、表达式、循环控制结构等知识将分别在2.2节和2.3节中进行讨论。

【练习 2.1】

请读者自行写出统计50名学生的成绩之和的步骤，并与引例比较。

2.2 基础知识

通过前面讨论的例题，无论是类、方法还是变量都必须有一个名称，不同的数据有不同的特征，以数据类型加以区分。程序设计中的计算表达式与数学公式的表示方式也不相同。本节

将逐一讨论这些内容。



视频

2.2.1 标识符

标识符就是编写程序时使用的各种字符序列，就如同数学上用 x 、 y 、 z 等来表示未知数一样。Java 对标识符的使用有严格的规定。标识符是赋予类、方法或者变量的名称。

1. 标识符的分类

标识符一般可分为以下三种类型。

- ❑ 程序设计人员自行选用的标识符；
- ❑ 其他程序设计人员选用的标识符，如 `String`、`System`、`out`、`println` 和 `main`；
- ❑ 语言中保留特殊含义的标识符，即关键字，如 `class`、`public`、`static` 和 `void`。

其他程序设计人员选用的标识符，是指已经被其他程序员选用并定义了具体含义的标识符，通常是预先定义好的 Java 标准类库的一部分。在编程时，仅能使用它们，而不能再为其定义新的含义。事实上，即使引用的并非 Java 标准类库中的类和方法，只要在程序中引用了其他程序员编制的类，一般也不建议使用所引用类中已经定义了的标识符。

程序员可以根据需要在程序中定义标识符。例如，例 2.1 中的 `Jpro2_1`、`count_score`、`sum`、`average_score`、`rate_score` 等，但不能使用保留字及已经被其他程序员选用的标识符。标识符可以由字母、数字、下画线（`_`）及美元符号（`$`）按一定的顺序组合而成，但不能以数字开头。例如，`average`、`table12` 及 `$price` 等均为有效的标识符，而 `5_step` 则为非法标识符。Java 语言标识符的命名规则如下。

- ❑ 标识符由字母、下画线、美元符号和数字组成，但不能以数字开头，长度不受限制；
- ❑ 严格区分字母的大小写，如 `Student` 和 `student` 是两个不同的标识符；
- ❑ 不能与 Java 关键字重名，不能与 Java 类库中的类名重名。

2. 关键字

关键字是指被系统保留使用并赋予特定意义的一些标识符，只能按照预先定义好的方式使用，不能被编程人员用作标识符，也不能作为其他用途。Java 的关键字对 Java 的编译器有特殊的意义，它们用来表示一种数据类型，或者表示程序的结构等。如果它们不是作为关键字出现在 Java 程序中，Java 编译器能够识别它们并产生错误信息。Java 语言有以下保留关键字，根据它们的意义分为以下几种类型。

- ❑ 数据类型：`boolean`、`int`、`long`、`short`、`byte`、`float`、`double`、`char`、`class`、`interface`。
- ❑ 流程控制：`if`、`else`、`do`、`while`、`for`、`switch`、`case`、`default`、`break`、`continue`、`return`、`try`、`catch`、`finally`。
- ❑ 修饰符：`public`、`protected`、`private`、`final`、`void`、`static`、`abstract`、`synchronized`、`strictfp`、`transient`、`volatile`、`native`、`var`。
- ❑ 动作：`package`、`import`、`throw`、`throws`、`extends`、`implements`、`this`、`super`、`instanceof`、`new`。
- ❑ 保留字：`true`、`false`、`null`、`goto`、`const`。

Java 中还有一类关键字，或者称预留关键字，它们虽然现在没有作为关键字，但在以后的升级版本中有可能作为关键字。例如，`cast`、`future`、`generic`、`inner`、`operator`、`outer`、`rest` 等都是保留字，在 Java 中也不能使用它们作为标识符。

2.2.2 Java 基本数据类型



视频

Java 中的数据类型分为基本数据类型和引用数据类型两类。Java 基本数据类型包括数值型、字符型和布尔型。引用数据类型包括类、接口和数组等，即通过对象的创建获得引用类型的值。

1. 数值型

Java 的数值型数据分为整数和浮点数两种类型，整数不含小数点，浮点数含有小数点。整数有 byte（字节型）、short（短整型）、int（整型）及 long（长整型）等 4 种，浮点型数据有 float（单精度浮点型）和 double（双精度浮点型）两种。表 2-1 显示了 Java 中所有的数值类型。

表 2-1 Java 中的基本数值类型

类 型	存 储 位 数	取值范围（十进制）	默 认 值
byte	8	-127~127	0
short	16	-32768~32767	0
int	32	-2147483648~2147483647	0
long	64	-9.2E+18~9.2E+18	0L
float	32	近似为-3.4E+38~-1.4E-45,1.4E-45~3.4E+38	0.0F
double	64	近似为-1.7E+308~-2.2E-208,2.2E-208~1.7E+308	0.0D

存储数据要占用一定的存储空间，不同类型的数据所占用的存储空间不同。所有数值类型依据其占用的内存空间大小进行区分。在设计程序的过程中，程序员需要选择大小合适的变量类型，否则有可能造成内存空间的浪费。

Java 认为所有的整数都是 int 型，只有在整数后面加一个 L 或者 l 才可以将其表示成 long 型值，如 69L、056L、0xfbL 等。同样，Java 认为所有的浮点值都属于 double 型的数据。如果需要使用 float 型的数据，需要在数值后面加一个 F 或者 f，如 3.669F 或者 6.223f 等，当然也可以通过在数值后面加上 D 或 d 表示 double 型，如 3.78d。

在实际应用中，特别是在给变量赋值时，应保证数值类型与变量类型的一致。下面的数据类型定义及赋值说明了这一点。

```
int num=3;
float fFloat=5.0f;
long len=0L;
```

当然，Java 有自动转换数值类型的机制，请参阅 2.2.3 节的相关内容。

对于整型数据，可用使用十进制表示，还可以用八进制、十六进制表示。八进制以 0 开头，例如 054 与 036。十六进制以 0x 开头，例如 0x78 及 0x3AFB。而对于浮点型数值也有十进制表示法与科学计数法两种。十进制形式由数字和小数点组成，例如，305.06、0.0045 等。科学计数法由数字和 e（或 E）组成，e（或 E）前必须有数字，后面是整数，例如，1234.567 可表示为 1.234567E3，0.00654 可表示为 6.54E-3。

2. 字符型

Java 中的字符型数据用 char 表示，它的值用 16 位（2 字节）来存储，取值范围是 0~65535。Unicode 字符集中的所有字符，根据 Unicode 编码，可以比较其大小，类似于 ASCII 的比较。与其他语言一样，Java 也用单引号来表示字符型数据。如'A'、'c'、'#'、'&'与'9'等。例如

```
char grade1='A', grade2='B';
```

显然, grade1 和 grade2 可以比较大小, 'A'的编码值小于'B'的编码值, 所以 grade1 小于 grade2。

3. 布尔型

布尔型 (boolean) 是一种表示逻辑值的简单数据类型。它的取值只能是常量 true 或 false 这两个值中的一个, 在存储器中占 8 位 (1 字节), 通常用于程序中的一些逻辑判断, 从而对程序的运行进行控制。例如, 根据成绩是否达到或超过及格线 60 分, 判断考试是否通过。

```
int grade;
boolean passOrNo;
if (grade>=60)
    passOrNo=true;
else
    passOrNo=false;
```



视频

2.2.3 常量和变量

在程序中经常要用到一些数据, 例如, 学生的年龄、成绩、性别、学号等, 这些值有的是已知的且固定不变的, 有的则随着环境或者状态的变化而变化。对这些数据进行处理就要定义相应的常量与变量, 并为其赋值。

1. 常量

常量 (constant), 就是在程序运行过程中其值不会被改变的量, 就像在数学中常用到的圆周率 $PI=3.14$ 。常量也叫常数。在 Java 中, 使用关键字 final 声明一个符号常量。其格式为:

```
final 数据类型 常量名称=常量值;
```

上式中的数据类型可以是 Java 中任意合法的数据类型, 如 int、float、double、char、String 等。常量名称必须是 Java 的合法标识符, 一般采用大写字母单词命名, 单词与单词之间用下划线加以分隔, 例如

```
static final int MAX_WIDTH = 1000;
static final double PI=3.14;
```

在 Java 中, 按照数据的特征, 可将常量分为整型、浮点型、字符型、字符串型与布尔型等 5 种类型。

整型常量和浮点型常量都属于数值型, 其分类及说明与 2.2.2 节中的介绍相同。

字符型常量是指 Unicode 字符集中的所有单个字符, 包括可以打印的字符和不可打印的控制字符, 它的表示形式有以下 4 种。

(1) 以单引号括起来的单个字符, 例如 'A'、'h'、'*'、'l'。

(2) 以单引号括起来的 \ 加三位八进制数, 形式为 '\ddd', 其中 d 可以是 0~7 中的任一个数, 例如, '\141' 表示字符 'a'。其中, ddd 的取值范围只能在八进制数的 000~777 中, 因而它不能表示 Unicode 字符集中的全部字符。

(3) 以单引号括起来的 \u 加 4 位十六进制数, 例如, '\u0061' 表示字符 'a'。这种表示方法的取值范围与 char 型数据相同, 因而可以表示 Unicode 字符集中的所有字符。

(4) 对于那些不能被直接包括的字符以及一些控制字符, Java 定义了若干转义序列。如 '\' 代表 '\', '\n' 代表换行等。表 2-2 所示的是 Java 中的一些字符转义序列。

表 2-2 Java 中的转义序列

转 义 序 列	含 义	转 义 序 列	含 义
\'	单引号	\f	换页
\"	双引号	\n	换行
\\	反斜杠	\r	回车键
\b	退格	\t	水平制表符

字符串常量是由用双引号括起来的零个或多个字符组成的字符序列，如"Hello World!"、"I am a programmer.\n"等。字符型常量的八进制、十六进制表示法和转义序列在字符串中同样可用。需要注意的是，'A'与"A"是不同的，前者是字符，后者是字符串。同样，12.345 与"12.345"也是不同的，读者应注意加以区分。字符串常量可以用 String 类来定义，String 类将在后续章节中详细讲述，读者在此只需把它当作字符串类型数据即可。

布尔型常量只有两个值：true 和 false。

2. 变量

变量（variable）是 Java 程序中的基本存储单元，是在程序运行过程中其值可以改变的量。一个变量包含有三层含义：①变量的名称，简称变量名，变量名是用户自己定义的标识符，它表明了变量的存在；②变量的属性，即变量的数据类型，包括基本数据类型和引用数据类型；③变量的值，是指存放在变量名所标记的存储单元中的数据。

Java 中所有的变量必须是先声明后使用。变量的声明方法是：

```
数据类型 变量名 1 [=初值 1] [, 变量名 2 [=初值 2] ...];
```

其中，数据类型必须是 Java 的基本数据类型之一，或者是类、接口类型的名称（类和接口将在第 3 章和第 5 章中讨论）。变量名可以是任意合法的 Java 标识符，变量名一般以小写字母开头，如果由多个单词构成，第一个单词的首字母小写，其后单词的首字母均大写。变量名的选用应该易于记忆，且能够指出其意义。方括号中的内容是可选项，用=号加初值给变量赋初值，例如

```
int math=78,english=80;
```

一个变量必须经过声明、赋值之后才能被使用；类型相同的几个变量可以在同一个语句中被声明及被赋初值，相互之间应用英文逗号（,）作间隔。

下面通过例子说明变量的定义与赋值。

【例 2.2】求三门课程的平均分。

```
//文件名 Jpro2_2.java
1 public class Jpro2_2{                                //定义主类
2     public static void main (String[] args) {        //定义 main() 方法
3         int pro_score,eng_score,math_score;          //声明变量
4         pro_score=89;                                //输入三门课程成绩
5         eng_score=86;
6         math_score=93;
7         System.out.println("平均分是："+(pro_score+eng_score+math_score)/3);
8     }
9 }
```

程序分析：程序中定义了三门课程成绩的变量类型均为整型。但程序运行有误差，因为该平均分为 89.333 的小数，而程序运行结果为 89。这是数据类型出问题了，因为三门课程成绩变量为

整数，除数 3 也是整数，求得的平均值数据类型不变，所以结果值为整数 89。要输出正确结果，需借助类型转换把平均值的数据类型变为小数，这就涉及下面所说的类型转换及格式设置。利用类型转换，把第 7 行程序修改成“System.out.printf("平均分是:%.1f", (float)(pro_score + eng_score + math_score) / 3);”，这样程序运行结果就正确。

3. 赋值语句

在前面已经使用了这样的语句，例如 `int a=10;`，它的意思是声明变量 `a` 是整型变量，并且给 `a` 赋初值为 10。这条语句与下面两条语句是等价的：

```
int a;
a=10;
```

第二条语句就是一条赋值语句。赋值语句的一般形式为：

变量名=表达式；

在上式中，变量名是一个已经被声明过的变量；表达式是一个可以计算出确定值的式子，但表达式的值必须与变量的数据类型兼容；语句的最后必须以分号结束。

赋值语句执行的过程中，先计算出表达式的值，然后把该值存储到赋值运算符(=)左边的变量所代表的存储单元，并覆盖其原有的值。例如

```
float value, price;
value=0.0f;           //value 初始化后值为 0.0
price=0.5f;
value=12*price;        //value 重新赋值后为 6.0
```

4. 类型转换

Java 是强类型语言，其中的每个数据都有特定的数据类型。Java 中所有的数值传递都必须进行类型相容性检查以保证类型是兼容的。任何类型不匹配都将被报告为错误。因此，在进行程序设计时经常要对一些类型不同的数据进行类型转换。Java 的数据类型转换有两种情况：表达式中的自动类型转换和强制类型转换。

(1) 表达式中的自动类型转换。

Java 表达式中，当涉及两个不同类型的数据进行运算时，系统会自动把两个不相同的类型数据转换成相同的类型再进行运算。这种转换是在程序运行过程中不需人为干预而自动进行的，但转换是有条件的，即两种数据类型必须是兼容的，规则是把表达式中取值范围小的类型数据转换成另一取值范围较大的数据类型。例如

```
int a;
float b;
double c;
```

若有表达式 `a+b+c`，先计算 `a+b`，`a` 被转换成 `float` 型与 `b` 相加，结果为 `float` 型；然后结果再被转换成 `double` 型与 `c` 相加，结果为 `double` 型数据。可自动进行转换的数据类型如表 2-3 所示。

表 2-3 自动类型转换关系表

源数据类型	目标数据类型	源数据类型	目标数据类型
byte	short、int、long、float、double	int	long、float、double
short	int、long、float、double	long	float、double
char	int、long、float、double	float	double

(2) 强制类型转换。

对于类型不一致数据,在写表达式时不能进行自动类型转换,这时就要执行强制类型转换。强制类型转换的一般格式为:

(目标数据类型) 被转换数据

被转换数据可以是变量或表达式等,如要把 double 型变量 price 的值转换成 int 型,形式为:(int)price。

把表 2-2 中的目标类型向源类型的转换都必须使用强制转换。这种转换是将取值范围较大的类型向取值范围较小的类型转换,但结果可能带来两个问题:精度损失和溢出。

例如,将浮点型数据转换为整型数据,其结果是小数部分丢失。

```
float a=123.45f;
int b;
b=(int)a;
```

在使用数据类型的强制转换时,读者要特别注意这两个问题。另外,不管是自动转换还是强制转换,转换的只是变量或表达式的“读出值”,而变量、表达式自身的类型和值均未被改变。

【例 2.3】 从键盘读入一个字符,并输出这个字符。

```
//文件名 Jpro2_3.java
1  import java.util.Scanner;
2  public class Jpro2_3 {
3      public static void main(String[] args) {
4          System.out.println("Input a character:");
5          Scanner scanner = new Scanner(System.in);
6          String input = scanner.next();           //读取一个字符串
7          char ch = input.charAt(0);               //取第一个字符
8          System.out.println("The input is '" + ch + "'");
9          scanner.close();
10     }
11 }
```

请读者自行运行程序并分析结果。

程序分析: 程序定义了一个字符型变量 ch,接收输入的字符。第 5 行创建 Scanner 对象,new Scanner(System.in)表示从标准输入流(键盘)读取数据。第 6 行调用 scanner.next()方法读取一个字符串。第 7 行通过 charAt(0)方法提取该字符串的第一个字符,并赋值给 ch。第 8 行将字符输出。第 9 行关闭 Scanner 对象。

2.2.4 运算符与表达式

前面已经用到了一些表达式,例如 a+b+c,就是一个算术表达式。其中,+号是一个运算符,a、b、c 本身也是一个表达式。一个常量或一个变量是最简单的表达式。表达式是由数据和运算符按 Java 语法规则组成的式子。这里的数据是常量或变量,表达式中数据的连接符+、-、*、=及<就是运算符。Java 的运算符主要包括算术运算符、关系运算符、逻辑运算符等。

1. 算术运算符和算术表达式

Java 中的算术运算符主要用来对整型及浮点型数据进行运算,也可以对字符型数据进行运算。算术运算符可以分为单目运算符和双目运算符。



视频

(1) 单目运算符。

单目运算符是指只对一个操作数运算的运算符。Java 中的单目运算符有++（自增）、--（自减）和-（取反）等三种类型。单目运算符++与--可以位于操作数的左边或右边，但在使用时是有差别的。例如：

- ❑ a++、a--：表示先使用 a，再使 a 增（减）1。
- ❑ ++a、--a：表示先使 a 增（减）1，再使用 a。

(2) 双目运算符。

双目运算符是指算术运算符的左、右两边均要有操作数。Java 中的双目算术运算符有+（加运算）、-（差运算）、*（积运算）、/（除运算）及%（求余运算）。

注意：两个整数相除时其结果仍是整数，小数部分被舍去。另外，%运算符既可以对整数进行操作也可以对浮点数进行操作。当除数为 0 时，/和%运算会产生异常，需要进行异常处理。

(3) 算术表达式。

算术运算符连接的操作数为数值型。由算术运算符连接的式子称为算术表达式。例如，++a; s=(a1+a2+a3)/3; totalPrice=(weight*unitPrice+tax)* discount。

2. 关系运算和逻辑运算

在程序中，一个运算的执行通常在某个条件下，根据条件是否满足判断运算能否执行。这个判断过程可以使用关系运算或逻辑运算。

(1) 关系运算符和关系表达式。

关系运算符是用来比较两个值之间的大小关系的。关系运算的结果是布尔（boolean）型，结果为真（true）或假（false）。Java 中的关系运算符如表 2-4 所示。

表 2-4 关系运算符

运 算 符	操 作	功 能
==	操作数1 == 操作数2	判断操作数1是否等于操作数2
!=	操作数1 != 操作数2	判断操作数1是否不等于操作数2
>	操作数1 > 操作数2	判断操作数1是否大于操作数2
<	操作数1 < 操作数2	判断操作数1是否小于操作数2
>=	操作数1 >= 操作数2	判断操作数1是否大于或等于操作数2
<=	操作数1 <= 操作数2	判断操作数1是否小于或等于操作数2

例如

```
float grade=70,pass=60;
boolean c=grade>=pass;
```

则结果 c=true。

(2) 逻辑运算符和逻辑表达式。

Java 中共有 6 个逻辑运算符。如果它们的操作数是布尔类型的数据，其结果也是布尔型。各逻辑运算符如表 2-5 所示。

表 2-5 逻辑运算符

运 算 符	名 称	例 子	功 能
!	逻辑非	!a	a为真时值为假，反之亦然
&	逻辑与	a&b	a、b均为真时值为真，否则为假
^	逻辑异或	a^b	a、b同值时值为假，异值时为真
	逻辑或	a b	a、b均为假时值为假，否则为真
&&	逻辑与	a&& b	a、b均为真时值为真，否则为假
	逻辑或	a b	a、b均为假时值为假，否则为真

表 2-5 中&和&&、|和||在形式上具有相同的名称及功能，其实它们是有差别的。&&和||也被称为“短路逻辑运算符”，在运算过程中会产生“短路效应”，如 a&&b，当 a 为 false 时不再判断 b，直接判定 a&&b 的值为 false；若是 a||b，当 a 为 true 时不再判断 b，直接判定 a||b 的值为 true。而对于&和|则不会出现这种情况，它们是先计算出两边操作数的值，然后再进行逻辑判断。另外，当&、^和|的操作数为布尔类型时它们是逻辑运算符，进行逻辑运算，但若操作数为整数及字符时，它们会作为位运算符进行位运算。

关系运算和逻辑运算通常又称为布尔运算，一般应用于条件语句中，在后续内容中会经常使用布尔运算。例如，判断某个整数 m 是否为偶数，条件语句表达式为 if(m%2==0)。

【例 2.4】 逻辑运算示例。

```
//文件名 Jpro2_4.java
1 public class Jpro2_4{
2     public static void main(String[] args) {
3         String account = "admin";
4         String password = "123456";
5         final String SUCCESS = "登录成功";
6         if (account.equals("admin") && password.equals("123456")){
7             System.out.println(SUCCESS);
8         }
9     }
10 }
```

请读者自行运行程序并分析结果。

【AI 提示词】

初学编程时，可借助 AI 助手逐行理解程序代码。请使用以下提示词与 AI 交流，让其引导你分析代码逻辑，而非直接给出运行结果。

(1) 请带着我逐行阅读以下程序。对每一行代码，请依次说明这行代码的含义。

(2) 执行该行后程序发生了什么变化。请用初学者易于理解的语言讲解，避免使用过于专业的术语。

下面是我需要理解的代码：

【在此处粘贴你的代码】

2.2.5 其他运算符

除 2.2.4 节介绍的运算符外，还有其他运算符，如三元运算符、位运算符及赋值运算符等。

1. 三元运算符

条件表达式由三元运算符构成，格式如下：

条件表达式?语句 1:语句 2

其中的?和:称为三元运算符，它们必须一同出现，此运算符需要三个操作数。其中，语句 1 和语句 2 可以是复合语句。其意思是，当条件表达式值为 true 时，执行语句 1，否则执行语句 2。

说明：

- ① 条件表达式运算符的优先级别很低，仅优先于赋值运算符。
- ② 条件运算符的结合性为自右向左。例如

`(a>b)?a:(c>d)?c:d`

其中 a=5, b=8, c=1, d=9。根据右结合性，应先计算(c>d)?c:d。因为 1>9 为 false，故取 d=9 为该表达式的结果。再计算(a>b)?a:d，则最终结果为 9。

2. 位运算符

Java 中，可以使用位运算直接对整数类型和字符类型的数据的位进行操作。Java 中的位运算符如表 2-6 所示。

表 2-6 位运算符

运 算 符	名 称	例 子	功 能
~	按位非	~a	a按位取反
<<	左移	a<<b	a左移b位，右边补零
>>	带符号右移	a>>b	a右移b位，若a的最高位为1，左边补1，否则补0
>>>	无符号右移	a>>>b	a右移b位，左边补0
&	按位与	a&b	a和b按位与
^	按位异或	a^b	a和b按位异或
	按位或	a b	a和b按位或

Java 中的数是以补码表示的。正数的补码就是其原码，负数的补码是其对应的正数按位取反（1 变为 0，0 变为 1）后再加 1。位运算的方法读者可以从下面的例子中仔细体会。例如，byte a=7, b=-7, c=15, d=42，则

a: 00000111	b: 11111001	
c: 00001111	d: 00101010	
~c= -16: 11110000		
a<<2=28: 00011100	b>>2= -2: 11111110	b>>>2=62: 00111110
c&d=10: 00001111	c^d=37: 00001111	c d=47: 00001111
00101010	^ 00101010	00101010
-----	-----	-----
00001010	00100101	00101111

3. 赋值运算符和赋值表达式

赋值运算符=前面已多次使用，相信读者已不陌生。例如，a=b 表示变量 b 的值赋给变量 a，则 b 与 a 的值相同，a 原来的值丢弃。赋值运算符两边数据类型可以不相同但必须相容。当数据类型不相同，若右边数据取值范围小于左边数据，则会自动转换；反之，则必须强制转换。除了上面的=之外还有一些扩展的赋值运算符。这些扩展赋值运算符的使用不仅可以使程序表达

简练,而且可以提高程序的编译速度。扩展赋值运算符就是把赋值运算符与算术运算符、逻辑运算符或位运算符中的双目运算符结合起来而形成的赋值运算符,如表 2-7 所示。

表 2-7 扩展赋值运算符

运 算 符	名 称	例 子	功 能
<code>+=</code>	加并赋值	<code>a+=b</code>	<code>a=a+b</code>
<code>-=</code>	减并赋值	<code>a-=b</code>	<code>a=a-b</code>
<code>*=</code>	乘并赋值	<code>a*=b</code>	<code>a=a*b</code>
<code>/=</code>	除并赋值	<code>a/=b</code>	<code>a=a/b</code>
<code>%=</code>	取余并赋值	<code>a%=b</code>	<code>a=a%b</code>
<code>&=</code>	按位(逻辑)与并赋值	<code>a&=b</code>	<code>a=a&b</code>
<code>^=</code>	按位(逻辑)异或并赋值	<code>a^=b</code>	<code>a=a^b</code>
<code> =</code>	按位(逻辑)或并赋值	<code>a =b</code>	<code>a=a b</code>
<code><<=</code>	左移并赋值	<code>a<<=b</code>	<code>a=a<<b</code>
<code>>>=</code>	带符号右移并赋值	<code>a>>=b</code>	<code>a=a>>b</code>
<code>>>>=</code>	无符号右移并赋值	<code>a>>>=b</code>	<code>a=a>>>b</code>

显然,用赋值运算符连接起来的式子就是赋值表达式。表 2-7 的第 3、第 4 列全部是赋值表达式。

4. 运算符优先级

Java 语言中的一个表达式包含了多个运算符,哪个运算符先运算哪个后运算,与数学中的运算规则类似,也要有规则,即要设置运算符的优先级。Java 语言运算符的运算级别共分 15 级,其中 1 级的优先级最高,15 级最低,如表 2-8 所示。

表 2-8 运算符的优先级

优 先 级	运 算 符	含 义	结 合 性
1	<code>[]</code> <code>.</code> <code>()</code>	数组下标、对象成员、计算及方法调用	从左到右
	<code>++</code> <code>--</code>	先用后增、先用后减	
2	<code>++</code> <code>--</code>	先增后用、先减后用	从右到左
	<code>+</code> <code>-</code>	正号、负号	
	<code>~</code> <code>!</code>	按位非、逻辑非	
3	<code>new</code> (类型)	对象实例化、强制类型转换	从右到左
4	<code>*</code> <code>/</code> <code>%</code>	乘、除、取余	从左到右
5	<code>+</code> <code>-</code>	加、减	从左到右
6	<code><<</code> <code>>></code> <code>>>></code>	左移、带符号右移、无符号右移	从左到右
7	<code><</code> <code><=</code>	小于、小于或等于	从左到右
	<code>></code> <code>>=</code>	大于、大于或等于	
8	<code>==</code> <code>!=</code>	相等、不等	从左到右
9	<code>&</code>	逻辑(按位)与	从左到右
10	<code>^</code>	逻辑(按位)异或	从左到右
11	<code> </code>	逻辑(按位)或	从左到右

续表

优 先 级	运 算 符	含 义	结 合 性
12	&&	逻辑与	从左到右
13		逻辑或	从左到右
14	?:	条件运算符	从右到左
15	=	赋值	从右到左
	扩展赋值符	扩展赋值	

例如，计算下面表达式的值：

```
int a=3,b=8,c;
c=a++b;
```

上述第二条语句执行后，变量 a、b、c 的值分别是多少呢？如果表达式被理解为 c=a+(++b)，那么，a=3、b=9、c=12。其实结果并不是这样，而是 a=4、b=8、c=11。这是因为运算符++作为“先用后增”的优先级高于作为“先增后用”运算符的优先级。在此建议初学者在写表达式时多使用括号，这样既能防止引起混乱，又方便他人阅读你的程序代码。

【例 2.5】 判断某一年是否为闰年。

```
//文件名 Jpro2_5.java
1 public class Jpro2_5{
2     public static void main(String args[]){
3         int year=2020;
4         boolean t;
5         t=year%400==0||year%4==0&&year%100!=0;
6         System.out.println(year+" a leap year =" +t);
7     }
8 }
```

程序运行结果如下：

```
2020 is a leap year =true
```

注意：第 7 行语句是一个赋值运算表达式，右边部分是逻辑表达式，其中包含算术运算、关系运算和逻辑运算，请读者给出该表达式的运算优先级，并分析结果。

程序分析：本程序每次运行只能判断一个年份是否为闰年，如果要判断另一个年份是否为闰年，则要修改语句 int year=2020; 中的数值，因此很不方便。如果采用从键盘接收数据的方法，将大大改善程序。将语句

```
int year=2020;
```

修改为下面的语句即可。

```
Scanner in=new Scanner(System.in);
int year=in.nextInt();
```

这样，如果要判断 2020 年是否为闰年，在程序运行输入参数时只需键入 2020，即可判断输入的年份是否为闰年。再运行一次，输入另一个年份，则可判断其他年份是否为闰年。

实际上使用键盘接收参数，仍不是一个好方法，尤其当判别连续的年份时。在 2.3 节中将通过学习循环语句更便捷地解决这个问题。

【练习2.2】

1. [单选题]_____是合理的标识符。

- A. i++ B. 5age C. \$y D. case

2. [单选题]对于以下代码:

```
int a,b=3;
```

以下_____是对的。

- A. a 被初始为 0, 而 b 是 3 B. 不能通过编译
C. a 不会被初始化, 而 b 是 3 D. a 和 b 的初始值都是 3
3. [单选题]MAX_LENGTH 是 int 型常量, 值为 99, 用_____简短语句定义这个变量。
A. int MAX_LENGTH=99; B. final int MAX_LENGTH=99;
C. final public int MAX_LENGTH=99; D. final INT MAX_LENGTH=99;
4. [单选题]以下代码片段之行后, num 的值是_____。

```
int num = (int)5.5;  
num %= 2;
```

- A. 0 B. 1 C. 2 D. 3

2.3 基本控制结构与实现

引例中已涉及程序设计中的三种基本结构: 顺序结构、选择结构和循环结构。在解决具体问题时, 通常采用选择语句、循环语句实现相应的算法。顺序结构程序的执行完全按照程序书写顺序执行, 是最简单的一种基本结构。在前面的章节中, 程序主要采用顺序结构, 本节将详细介绍选择结构和循环结构的编程方法。

2.3.1 选择结构

选择结构根据给定条件的真假决定程序的执行分支。Java 提供了以下选择语句: 单分支 if 语句、双分支 if...else 语句、嵌套 if 语句、if...else if 语句、多分支 switch 语句等。

1. 单分支if语句

单分支 if 语句格式如下:

```
if (布尔表达式)  
{语句块;}
```

程序执行流程图如图 2-1 所示。

执行过程是, 如果布尔表达式值为 true, 则执行语句块; 否则, 不执行语句块, 程序执行流程转至 if 后面的语句。例如

```
if (x>0)  
y=1;
```

表示当 x>0 的值为 true 时, 执行语句 “y=1;”。

说明:

① 布尔表达式可以是布尔类型的常量、变量、关系表达式或逻辑表达式等, 如果是其他类型, 则编译出错。布尔表达式必须写在括号中。

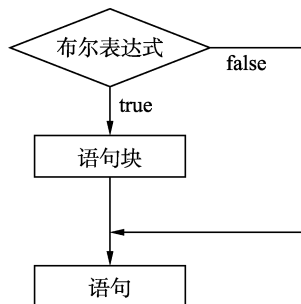


图 2-1 单分支 if 语句流程图



视频

② 语句块的语句可以是 Java 中的任何语句，若只有一条语句，可以省略{}；若包含多条语句，则必须使用{}。

例如，定义“int x=-5;”，在以下两个程序段中，x>0 的值为 false，if 语句块均不执行。

程序段一：

```
int y=0;
if(x>0)
    y=1;
System.out.println("y="+y);
```

程序段二：

```
int y=0;
if(x>0)
{
    y=1;
    System.out.println("y="+y);
}
```

在程序段一中，单语句“y=1;”是 if 语句块，该语句不执行；y 的值仍然为 0，跳出 if 语句后执行输出语句，输出结果为：

```
y=0
```

在程序段二中没有输出结果，请读者自行分析。

【例 2.6】 从键盘输入一个整数，判断该整数是否为偶数。

分析：判断输入的整数是否能被 2 整除，设置判断的布尔表达式为 num%2==0。若条件成立，则输出该数是偶数的提示信息。

程序源代码如下。

```
//文件名 Jpro2_6.java
1  import javax.swing.JOptionPane;           //引入类 JOptionPane
2  public class Jpro2_6{
3  public static void main(String[] args) {
4      String intString=JOptionPane.showInputDialog(null,"请输入一个整数:",
        "例 2.6", JOptionPane.QUESTION_MESSAGE);    //从键盘输入数据
5  int num=Integer.parseInt(intString);          //将 intString 转换成 int 型
6  if(num%2==0)                                  //判断 num 是否为偶数
7      System.out.println(num+"是偶数。");
8  }
9  }
```

程序运行后首先出现图 2-2 所示的对话框，如果输入 240，则输出“240 是偶数。”。

注意：本例介绍了另一种输入数据的方法，读者可以根据题目需要选择应用。

2. 双分支if…else语句

单分支 if 语句在指定条件为 true 时执行语句，否则不执行任何操作。如果要执行双分支选择操作，可以应用双分支 if…else 语句来实现。if…else 语句的格式如下：

```
if(布尔表达式)
{
    语句块 1;
}
else
{
    语句块 2;
}
```

程序执行流程图如图 2-3 所示。



图 2-2 例 2.6 输入对话框

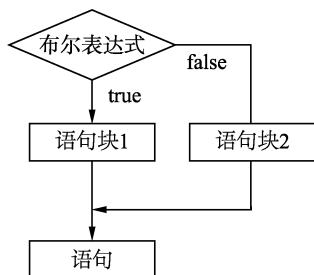


图 2-3 if...else 语句流程图

执行过程是，如果布尔表达式值为 true，执行语句块 1；否则执行语句块 2。

例如，下面是计算一个数的绝对值的函数：

$$y = \begin{cases} x & (x \geq 0) \\ -x & (x < 0) \end{cases}$$

可以用以下的程序段实现：

```
if (x >= 0)
    y = x;
else
    y = -x;
```

注意：以上函数也可以用下面的条件表达式实现：

```
y = (x >= 0) ? x : -x
```

【例 2.7】 用 Math 类的 random() 方法产生一个 0~1 的实数。若该数大于或等于 0.5，则输出 “num>=0.5”，否则输出 “num<0.5”。

分析：random() 方法产生的随机数在 0.0 和 1.0 之间。判断该数是否大于或等于 0.5，此处设置判断的布尔表达式为 num>=0.5。若布尔表达式值为 true，则输出 “num>=0.5”；若布尔表达式值为 false，则输出 “num<0.5”。

程序源代码如下。

```
//文件名 Jpro2_7.java
1 public class Jpro2_7{
2     public static void main(String args[]){
3         double num;
4         num=Math.random();           //产生一个随机数，关于该方法的具体应用请参考
                                         //Math 类相关 API
5         if(num>=0.5)
6             System.out.println("num>=0.5");
7         else
8             System.out.println("num<0.5");
9     }
10 }
```

请读者自行分析程序结果。

3. 嵌套 if 语句

if 语句或 if...else 语句中的语句块可以是任何合法的 Java 语句，包括 if 或 if...else 语句。这

种结构称为嵌套 if 语句。嵌套可以逐层展开，原则上没有深度的限制，但是嵌套的层数不宜过多。例如，下面就是一个嵌套的 if 语句。

```
if(score1>80){
    if(score2>80)
        System.out.println("score1 和 score2 都大于 80。");
    else
        System.out.println("score1 大于 80, score2 小于或等于 80。");
}
```

嵌套的 if 语句可以实现多重选择。例如下面的程序段，要求根据加、减、乘、除运算符计算表达式的值。

```
int x=5,y=4,z;
char ch='+';
if(ch=='+')
    z=x+y;
else if(ch=='-')
    z=x-y;
else if(ch=='*')
    z=x*y;
else if(ch=='/')
    z=x/y;
System.out.println(z);
```

执行过程是，从第一个 if 语句开始依次判断布尔表达式的值，当出现某个值为 true 时，则执行其对应的语句；如果所有的布尔表达式的值为 false，则执行 else 后的语句。只要一个条件满足，执行相应语句后 if 语句就结束，而不再对后面的布尔表达式进行判断。如果当前运算符是+，则进行加法运算。以上 if…else if 语句可以实现多重条件选择。

4. switch语句

过多使用嵌套的 if 语句，会增加程序阅读的困难，Java 提供了 switch 语句来实现多重条件选择。switch 语句根据表达式（整型或字符型）的值来选择执行多分支语句，一般格式如下：

```
switch (整型或字符型表达式) {
    case 常量表达式 1: 语句序列;[break;]
    case 常量表达式 2: 语句序列;[break;]
    :
    case 常量表达式 N: 语句序列;[break;]
    [default: 语句序列;]
}
```

执行过程是，计算整型或字符型表达式的值，并依次与 case 后的常量表达式值相比较，当二者值相等时，即执行其后的语句。

说明：

- ① 整型或字符型表达式必须为 byte、short、int、char 或 String 类型。
- ② 每个 case 语句后的常量表达式值必须是与表达式类型兼容的一个常量（它必须为一个常量，而不是变量），不允许出现重复的 case 值。
- ③ 关键字 break 为可选项，通常放在 case 语句块的末尾。执行此语句后，将终止当前 switch 语句。若没有 break 语句，将继续执行下面的 case 语句，直到 switch 语句结束或者遇到 break 语句。

④ default 为可选项。当指定的常量表达式都不能与 switch 表达式的值匹配时, 将选择执行 default 后的语句序列。case 语句和 default 语句次序无关, 但习惯上将 default 语句放在最后。

【例 2.8】 将上述根据加、减、乘、除运算符计算表达式值的问题用 switch 语句实现。程序源代码如下。

```
//文件名 Jpro2_8.java
1  import java.io.*;
2  public class Jpro2_8{
3      public static void main(String[] args) throws IOException{
4          int x=5,y=4,z=0;
5          char ch;
6          BufferedReader buf;                //声明缓冲字符输入流对象 buf
7          buf=new BufferedReader(new InputStreamReader(System.in));
8          ch=(char)buf.read();
9          switch (ch){
10             case '+': z=x+y;System.out.println(x+"+"+y+"="+z); break;
11             case '-': z=x-y;System.out.println(x+"-"+y+"="+z); break;
12             case '*': z=x*y;System.out.println(x+"*"+y+"="+z);break;
13             case '/': z=x/y;System.out.println(x+"/"+y+"="+z);break;
14             default: System.out.println("输入的运算符不符合要求!");
15         }
16     }
17 }
```

请读者自行运行程序并分析结果。

2.3.2 循环结构

循环结构是程序设计中实现重复操作的一种结构。其特点是, 在给定条件成立时, 反复执行某程序段, 直到条件不成立才终止。给定的条件称为循环条件, 反复执行的程序段称为循环体。Java 提供了三种形式的循环结构: while 循环、do...while 循环以及 for 循环。



视频

1. while 循环

while 循环又称当型循环, 它的格式如下:

```
while(布尔表达式){
    循环体;
}
```

执行过程是, 判断布尔表达式的值, 当其为 true 时, 执行循环体; 当布尔表达式的值为 false 时, 循环结束。while 循环的程序执行流程如图 2-4 所示。

例如, 下面的程序段在同一行输出 10 个 A。

```
int i=1;
while(i<=10){
    System.out.print ("A");
    i++;
}
```

上述程序段的循环条件是布尔表达式 $i \leq 10$, 循环体是 {} 中的两条语句。程序段执行时根据 i 的值判断 $i \leq 10$ 的值是否为 true, 若为 true 则执行循环体; 否则退出循环。循环体被执行 10 次, 当 $i=11$ 时, 退出循环。

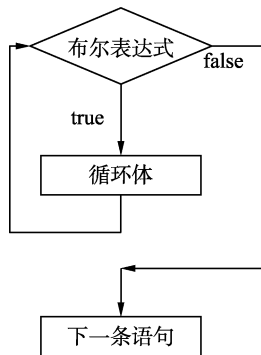


图 2-4 while 循环程序执行流程图

说明：

- ① 若循环体语句为单语句，{}可以省略；否则，不能省略{}。
- ② 若首次执行时循环条件为 false，则循环体一次也不执行；若循环条件永为 true，则循环体一直执行，这称为死循环。在循环体中应包含使循环结束的语句，以避免死循环。
- ③ 允许 while 语句的循环体也是 while 语句，从而形成循环的嵌套。

【例 2.9】 计算 $1-2+3-4+\cdots-100$ 。

分析：有规律的数据计算通常使用循环实现。程序中定义整型变量 sum 和 i，其中 sum 用于存放和，初值为 0，i 用作循环控制变量及其第二个加数，初值为 1。该算式中奇数项是加法，偶数项是减法，可以给第二个加数乘以标志变量来实现各项运算符的变化。程序源代码如下。

```
//文件名 Jpro2_9.java
1  public class Jpro2_9{
2  public static void main(String args[]){
3      int i,sum = 0;
4      int flag=1;                      //设置标志变量，控制运算符
5      i=1;                             //设循环初值
6      while(i<=100){                  //设循环条件为 i<=100
7          sum+=flag*i;
8          i++;                        //在循环体中执行 i++
9          flag=flag*(-1);
10     }
11     System.out.println("1-2+3-4+...-100="+sum);
12 }
13 }
```

程序运行结果如下：

```
1-2+3-4+...-100=-50
```

【AI 提示词】

如果对程序运行结果不理解，可借助 AI 助手重点讲解关键部分的执行过程。请使用以下提示词与 AI 交流：

请带着我逐行阅读以下程序。对每一行代码，用通俗易懂的语言讲解其含义和执行后发生的变化。如果遇到以下情况，请做特别说明：

对于循环结构，请重点讲清楚循环的执行顺序，包括循环条件的判断时机、循环体的执行次数以及退出循环的条件；

对于条件分支，请说明每个分支的触发条件和执行路径。

下面是我需要理解的代码：

【在此处粘贴你的代码】

2. do...while 循环

do...while 循环语句的特点是先执行循环体，再判断循环条件是否成立，格式如下：

```
do{
    循环体;
}while(布尔表达式);
```

执行过程是，首先执行一次循环体语句，然后判断布尔表达式的值，当其值为 true 时，返回重新执行循环体语句。如此反复，直到表达式的值为 false，循环结束。其流程如图 2-5 所示。

do...while 循环首先执行循环体,再判断循环条件。如果条件成立,则重复执行循环体;条件不成立,则结束循环,循环体至少被执行一次。而 while 循环首先判断循环条件,若条件不成立,则循环体一次也不执行,直接退出循环。这是 do...while 循环和 while 循环最大的区别。

do...while 循环语句可以组成多重循环,而且也可以和 while 语句相互嵌套。

【例 2.10】 用 do...while 循环计算 $1-2+3-4+\dots-100$ 。

程序源代码如下。

```
//文件名 Jpro2_10.java
1 public class Jpro2_10{
2     public static void main(String[] args){
3         int flag=1;
4         int i,sum=0;
5         i=1;
6         do{
7             sum+=flag*i;
8             i++;
9             flag=flag*(-1);
10        }while(i<=100);
11        System.out.println("1-2+3-4+...-100="+sum);
12    }
13 }
```

请读者自行运行程序并分析结果。

注意: 在 do...while 语句的 while(表达式)后必须加分号。

通过上面两例可以看出,对同一个问题可以用 while 循环语句处理,也可以用 do...while 循环语句处理。一般情况下,可以用两种语句处理同一问题,但要注意循环控制条件有些情况下可能不同。

3. for 循环

for 循环的使用最为灵活,可以用于循环次数已经确定的情况,也可以用于循环次数不确定但循环结束条件已知的情况。它可以取代 while 循环和 do...while 循环。for 循环语句的一般格式如下:

```
for (表达式 1;表达式 2;表达式 3) {
    循环体;
}
```

说明:

① ()内的三个表达式之间用分号分隔。其中,表达式 1 是 for 循环的初始化部分,一般用来设置循环控制变量的初值。表达式 1 允许并列多个表达式,之间用逗号分隔,表达式 1 仅在循环开始时执行一次;表达式 2 一般为条件表达式,结果为布尔型,当值为 false 时,退出循环,当值为 true 时,则重复执行循环体。表达式 3 一般是增量表达式,该表达式决定循环控制变量的

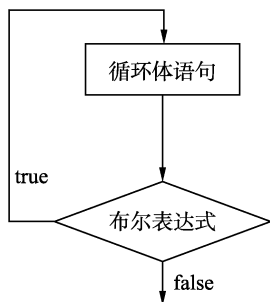


图 2-5 do...while 循环程序执行流程图

变化方式。

② 每执行循环体一次，就要重新计算表达式 3，然后由表达式 2 判断，决定循环体是否继续执行。循环体若为一条语句，{} 可以省略；若为多条语句，{} 不可省略。

【例 2.11】 用 for 循环计算 $1-2+3-4+\cdots-100$ 。

程序源代码如下。

```
//文件名 Jpro2_11.java
1 public class Jpro2_11{
2     public static void main(String[] args){
3         int flag=1;
4         int sum=0;
5         for(int i=1;i<=100;i++){
6             sum+=flag*i;
7             flag=flag*(-1);
8         }
9         System.out.println("1-2+3-4+...-100="+sum);
10    }
11 }
```

请读者自行运行程序并分析结果。

注意：for 循环、while 循环以及 do...while 循环都允许嵌套，并且可以相互嵌套，构成多重循环结构。

for 循环的使用方式比较灵活，可以有以下几种形式。

① 在 for 语句的表达式 1 中，允许定义多个变量，这些变量的数据类型相同并且它们的作用域仅限于循环体内。例如，计算 5!。

```
for(int i=1,p=1;i<=5;i++){
    p*=i;
```

② 在 for 语句中可以省略表达式 1。例如，计算 5!。

```
int i=1,p=1;
for(;i<=5;i++){
    p*=i;
```

③ 在 for 语句中可以省略表达式 2，不对循环条件进行判断，将会造成无限循环。一般可以采用在循环体内设置转移语句 break 来跳出循环。例如，计算 5!。

```
int p=1;
for(int i=1;;i++){
    if(i>5)break;                //结束循环
    p*=i;
}
```

循环体内的 if(i>5)break; 语句表示，当 i 的值大于 5 时，则跳出循环，它代替了原来表达式 2 所起的作用。

④ 在 for 语句中可以省略表达式 3。例如，计算 5!。

```
int p=1;
for(int i=1;i<=5;){
    p*=i;
    i++;                //改变循环变量 i 的值
}
```

⑤ for 语句中的各表达式都可以为空，但分号不能少。例如

```
for(;;)
```

在这种形式中，循环体内外应有相关语句实现各表达式的功能。

⑥ for 循环体可以是空语句，即循环过程什么也不做，仅仅产生一个时间延迟的效果。例如

```
for(int i=10;i>=1;i--);
```

2.3.3 控制转移语句

控制转移可以有条件地改变程序的执行顺序。Java 支持三种控制转移语句：break 语句、continue 语句和 return 语句。

1. break语句

break 语句的作用是使程序的执行流程从一个语句块内部转移出去。它只在 switch 语句和循环语句中使用，允许从 switch 语句的 case 子句中跳出，或从循环体内跳出。

break 语句分为带标号和不带标号两种形式：

```
break;
break 标号名;
```

其中，标号名为合法的 Java 标识符，用来标识 break 语句要跳出的语句块，它必须位于 break 语句所在的封闭语句块的开头处；标号名用冒号与其后面的语句分开。

带标号的 break 语句可以从多重循环的最内层跳出所有循环，而不带标号的 break 语句只能跳到当前循环外层。

【例 2.12】 break 语句应用实例。

程序源代码如下。

```
//文件名 Jpro2_12.java
1 public class Jpro2_12{
2     public static void main(String args[]){
3         for(int i=0; i<10; i++) {
4             if(i == 5) break;           //若 i 为 5，则终止循环
5             System.out.print(" i=" + i);
6         }
7     }
8 }
```

程序运行结果如下：

```
i=0 i=1 i=2 i=3 i=4
```

注意：程序中，当 i 的值为 0、1、2、3、4 时，i==5 的值为 false，不执行 break 语句，而是执行输出语句；当 i 的值变化到 5 时，i==5 的值为 true，执行 break 语句而中断循环，循环结束。本程序循环体共执行 5 次。

2. continue语句

continue 语句只能用在循环语句中，它的作用是终止当前这一轮循环，跳过本轮剩余的语句，直接进入下一轮循环。continue 语句具有带标号和不带标号两种形式：

```
continue;
continue [标号名];
```



视频

标号名必须放在循环语句之前，用于标志这个循环体。在 while 和 do...while 循环中，不带标号的 continue 语句使程序流程直接跳到循环条件的判断上；在 for 循环中，不带标号的 continue 语句直接计算表达式 3（增量表达式）的值，再根据表达式 2 的值决定是否继续循环。

【例 2.13】 continue 语句应用实例。

程序源代码如下。

```
//文件名 Jpro2_13.java
1 public class Jpro2_13{
2     public static void main(String[] args){
3         for(int i=0;i<10;i++){
4             if(i==5)continue;
5             System.out.print(" i="+i);
6         }
7     }
8 }
```

程序运行结果如下：

```
i=0 i=1 i=2 i=3 i=4 i=6 i=7 i=8 i=9
```

注意：程序中，当 i 的值为 0、1、2、3、4 时，i==5 的值为 false，不执行 continue 语句，而是执行输出语句；当 i 的值变化到 5 时，i==5 的值为 true，执行 continue 语句，跳过输出语句，然后转向执行 for 语句中的 i++，开始下一轮循环。

请读者注意例 2.12 和例 2.13 的区别。

3. return 语句

return 语句用在某个方法中。当程序执行到这条语句时，终止当前方法的执行，返回到调用这个方法的位置之后。

return 语句有带参数和不带参数的两种形式：

```
return;
return(表达式);
```

带参数的形式也可以为：

```
return 表达式;
```

不带参数的 return 语句被执行时，不返回任何值。这种方法的返回值类型为 void 类型。在没有返回值的方法体中，也可以省略 return 语句，程序执行完方法体的最后一条语句后，遇到方法的结束标志}时，程序流程将自动返回到调用这个方法的程序中。

带参数的 return 语句后面跟一个表达式，当程序执行到这个语句时，将计算该表达式的值，然后将其值返回到调用该方法的程序中。当表达式值的数据类型与方法的数据类型不统一时，以方法的返回值类型作为返回值类型。

关于 return 语句的应用在后续内容中将有多处涉及。

【练习2.3】

1. switch 后的表达式的值是哪些数据类型？使用 switch 语句需要注意什么？
2. while 语句和 do...while 语句之间的区别是什么？
3. 下列程序段执行后，k 的值是_____。

```
int x=3,y=5,k=0;
switch(x%y+3){
```

```

    case 0: k=x*y;break;
    case 6: k=x/y;break;
    case 12: k=x-y;break;
    default : k=x*y-x;break;
}

```

- A. 12 B. 0 C. 15 D. -2

4. 下面程序的执行结果是_____。

```

public class A{
    public static void main(String[] args) {
        int a=3;
        int b=4;
        int x=5;
        if(a*a+b*b==x*x)
            x=x<<(b-a);
        System.out.print(x);
    }
}

```

- A. 5 B. 6 C. 10 D. 3

5. 下列程序执行之后, 将会输出_____。

```

public class A{
    public static void main(String[] args) {
        int j=0;
        for(int i=3;i>=0;i--)
        {
            j+=i;
            System.out.print(j);
        }
    }
}

```

- A. 3566 B. 3522 C. 6753 D. 3255

6. 下列语句执行之后, j 的值是_____。

```

int j=9,i=6;
while(--i!=3)
    j=j+2;

```

- A. 9 B. 11 C. 13 D. 15

7. 下列语句执行之后, j 的值是_____。

```

int j=0;
for(int i=5;i>0&j<10;i--)
    j+=i;

```

- A. 9 B. 10 C. 11 D. 12

2.4 使用数组



在例 2.1 的中学生成绩存储在数组中, 如果用基本变量存储这 50 个学生数据, 则需要 50 个变量, 这样的程序健壮性和可移植性非常差。而引例中使用一个数组变量便可存储 50 个可独立访问的数据, 大大提高了程序的效率。这就是程序设计语言中引入数组的主要原因。



视频

2.4.1 创建数组

数组是 Java 中一种引用数据类型，它是由类型相同的数据组成的有序数据集合。集合中的每个数据是一个数组元素。在一个数组中

- ❑ 每个数组元素的数据类型都是相同的，在数组声明时定义；
 - ❑ 在内存中数组的各个元素是连续有序的；
 - ❑ 所有元素共用一个数组名，利用数组名和下标唯一地确定数组中每个元素的位置。
- 数组要经过声明、分配内存以及赋值后才能被使用。

1. 声明数组

数组的声明方式如下。

```
数据类型[] 数组名；
```

数组类型既可以是基本数据类型，也可以是引用数据类型，甚至还可以是其他数组类型数据。数组名命名规则和变量名相同，遵循标识符命名规则。

以下是几个数组声明的例子：

```
int[] array_int;
double[] array_double;
char[] str1;
String[] str2;
```

以上语句分别声明了一个整型数组 `array_int`、一个双精度型的数组 `array_double`、一个字符数组 `str1`、一个 `String` 类数组 `str2`，`str2` 中的每个元素可存放一个字符串。

2. 创建数组

数组中元素的个数称为数组大小或数组长度。与其他高级语言不同的是，Java 在声明数组时不能指定它的长度，必须通过创建数组来指定长度。可用以下方法来创建，利用关键字 `new` 为数组分配内存，即创建数组。创建数组的格式如下：

```
数组名=new 数据类型[数组长度]；
```

例如，`array=new int[10];`。

这条语句创建了一个名为 `array` 的数组，该数组有 10 个数组元素，为 `int` 型。

数组被创建以后，每个数组元素将获得与定义的数据类型相应大小的内存，同时自动用数据类型的默认值初始化所有的数组元素。各数据类型的默认值为：

- ❑ 整型：0。
- ❑ 浮点型：0.0f 或 0.0D。
- ❑ 字符型：'\0'。
- ❑ 类对象：null。

创建数组之后不能改变数组的大小。使用数组的 `length` 属性可以获得其长度，格式如下：

```
数组名.length
```

例如，`array.length` 的值为 10。

如果声明了一个数组但没有用 `new` 开辟空间，则数组不指向任何内存空间，其值为 `null`。声明一个数组并开辟内存空间后，则数组名是指向该内存空间的首地址。

声明数组变量以及创建数组可以组合在一条语句中，有下面两种形式。

(1) 第一种形式:

```
数据类型[] 数组名=new 数据类型[数组长度];
```

例如, `int[] array_int=new int[10];` 创建了包含 10 个 `int` 元素的数组, 每个元素默认值为 0。

(2) 第二种形式:

```
数据类型[] 数组名={值1, 值2, ...};
```

例如, `int[] array_int = {1, 3, 5, 7, 9};` 创建了一个数组, 其各个元素在内存中连续分配, 因此可以通俗地比作一次定义了多个变量, 并通过数组名[下标]访问, 下标从 0 开始。

3. 数组的赋值及引用

数组声明和创建后, 为获得有意义的初始值, 需对数组元素进行赋值。数组赋值有以下两种形式。

(1) 声明数组的同时初始化数组。

因为数组是引用类型, 它们的赋值与基本类型变量的赋值是不同的。

如果已知一组数据元素序列, 并要保存到数组中, 可以将这一组有序的元素放在花括号中, 各元素之间用逗号隔开, 并将其赋给数组, 实现声明数组的同时进行赋值。格式如下:

```
数据类型[] 数组名={第一个元素,第二个元素,第三个元素,...};
```

例如

```
int[] a={1,2,3,4,5};
float[] b={1.0f,2.0f,3.0f,4.0f,5.0f};
char[] c={'q','w','e','r','t'};
```

以上语句分别声明并创建了 `int` 型数组 `a`、`float` 型数组 `b` 和 `char` 型数组 `c`, 花括号中是各数组相对应元素的值。

(2) 先声明并创建数组, 再分别赋值。

例如, “`int[] array1=new int[10];`” 声明了包含 10 个元素的整型数组 `array1` 并为该数组分配了存储空间, 然后可通过赋值语句给数组中的各个元素赋值。例如

```
array1[0]=6;
array1[1]=7;
array1[2]=8;
```

给数组元素 `array1[0]`、`array1[1]` 和 `array1[2]` 分别赋值为 6、7、8。

数组通过以上两种形式被赋值后, 就可以被引用了。实际上, 在创建数组时, 数组元素已被初始化为数据类型的默认值了, 但通常需要赋予有实际意义的值。因此, 一般需要重新赋值。

数组元素通过其下标引用, 下标从 0 至数组名.length-1。引用格式为:

```
数组名[下标];
```

例如, `array1[0]` 表示引用数组 `array1` 中的第一个元素, `array1[1]` 表示引用数组 `array1` 中的第二个元素。下标必须为一个整数或一个整型表达式。引用数组元素时, 方括号中的下标不能超出它的取值范围。下面是一个数组引用的程序段。

```
for(int i=0;i<a.length;i++)
    System.out.println(a[i]);
```

上面 `for` 循环实现的功能是依次输出数组 `a` 中每个元素的值。

通常, Java 会自动进行数组下标越界检查, 如果下标超出该范围, 会产生 `ArrayIndexOutOf`

BoundsException 异常，即数组下标越界异常。因此，编写程序时最好使用数组的 length 属性获得数组大小，从而使下标不超出其取值范围。程序编译时，数组下标越界没有错误提示，但当程序运行时会产生运行错误。

【例 2.14】 在引例 2.1 的基础上，要求输出所有同学成绩，并输出该课程的最高分和最低分。

分析：输出数组元素是一个重复操作，可以用 for 循环实现，应用 println() 和 print() 方法可以控制输出格式。求最小值可以用以下算法实现：定义变量 min 存放最低成绩，定义变量 max 存放最高成绩，先将 pro_score[0] 的值赋给 min，然后 min 分别与 pro_score[i] (i=1 至 pro_score.length-1) 比较。若 pro_score[i] 小于 min，则将 pro_score[i] 的值赋给 min，否则 min 不变。比较操作完成后，min 中存放的数值即为数组 pro_score 中的最小值，即最低成绩。求最高成绩的方法同上（下面代码中没有输入所有同学的成绩）。

程序源代码如下。

```
//文件名 Jpro2_14.java
1  import java.util.*;
2  public class Jpro2_14{
3  public static void main(String args[]){
4      int i,min,max;
5      int[] pro_score=new int[50];           //声明定义数组
6      Scanner reader;                       //声明输入流类对象 reader
7      reader=new Scanner(System.in);
8      max=min=pro_score[0]=reader.nextInt();
9      for(i=1;i<pro_score.length;i++){      //定义循环体
10         if(pro_score[i]<min)
11             min=pro_score[i];
12         if(pro_score[i]>max)
13             max=pro_score[i];
14     }
15     System.out.println();
16     System.out.println("最高成绩是："+max);
17     System.out.println("最低成绩是："+min);
18 }
19 }
```

请读者自行运行程序并分析结果。



视频

2.4.2 字符数组

字符数组中的每个元素都是字符类型的数据，它的创建方法与其他类型的数组相似。

1. 字符数组的声明和创建

字符数组创建的格式如下：

```
char[] ch=new char[10];
```

上述语句声明并创建了字符数组 ch，该数组中可以存储 10 个字符。

字符数组可以初始化，例如

```
char ch={'a', 'b', 'c', 'd', 'e'};
```

其中，ch 是一个字符数组，共有 5 个元素，ch[0] 为 'a'，ch[1] 为 'b'，ch[2] 为 'c'，ch[3] 为 'd'，

ch[4]为 'e'。

字符数组的元素可以被赋值，例如

```
ch[1]='d';
```

下面通过 for 循环语句给一个字符数组赋值。

```
char[] ch=new char[100];
for(int i=0;i<26;i++)
    ch[i]='A'+i;
```

经过上述循环后，数组 ch 中将存储 26 个大写字母。

2. 字符串与字符数组

字符串不是字符数组，但是可以转换为字符数组，反之亦然。字符串和字符数组之间的转换主要有以下两种形式。

(1) 使用 toCharArray 方法将字符串转换为字符数组。

例如，下面语句将字符串 "word" 中的字符转换为数组 charArray 中的数组元素。

```
char[] charArray="word".toCharArray();
```

则 charArray[0]是'w'，charArray[1]是'o'，charArray[2]是'r'，charArray[3]是'd'。

(2) 使用 String(char[])构造方法或 valueOf(char[])方法将字符数组转换为字符串。

例如，下面语句使用 String 类的构造方法由数组构造一个字符串。

```
String str=new String(new char[]{'w','o','r','d'});
```

下面语句使用 valueOf 方法由数组构造一个字符串。

```
String str=String.valueOf(new char[]{'w','o','r','d'});
```

关于此知识点的详细介绍请参考常用实用类中字符串相关的 API。

【例 2.15】 编写程序判断输入的字符串是否为回文。

分析：回文是指一个字符串的顺序和逆序相同，例如 mom 是回文。判断是否是回文的方法很多，本例采用以下算法实现：将字符串转换到一个字符数组，检查字符数组的第一个元素是否和最后一个元素相同。如果相同，则继续检查第二个元素是否和倒数第二个元素相同。这个过程持续到检查出不相同的元素或字符串中的所有字符都已检查完。一旦检查过程中出现不相同的元素，则该字符串不是回文；若所有元素均比较完且未出现不相同元素，则该字符串为回文。

程序源代码如下。

```
//文件名 Jpro2_15.java
1 import java.util.Scanner;
2 public class Jpro2_15 {
3     public static void main(String[] args) {
4         Scanner scanner = new Scanner(System.in);
5         System.out.print("请输入一个字符串: ");
6         String s = scanner.nextLine();
7         if (isPalindrome(s))
8             System.out.println(s + "是回文");
9         else
10            System.out.println(s + "不是回文");
11        scanner.close();
12    }
13 }
```

```

14     public static boolean isPalindrome(String s) {
15         int low = 0;
16         int high = s.length() - 1;
17         while (low < high) {
18             if (s.charAt(low) != s.charAt(high))
19                 return false;
20             low++;
21             high--;
22         }
23         return true;
24     }
25 }

```

请读者自行运行程序并分析结果。



视频

2.4.3 遍历数组

数组指的是一组相同类型的变量集合，并且这些变量可以按照统一的方式进行操作。数组遍历在项目开发中经常用到，依次访问数组中的每个元素（部分有 if 条件+break 的则可能不会完全遍历），可以对数组中的每个数进行处理，或找到数组中需要的数。

使用标准的 for 循环可以完成一个数组的遍历。

【例 2.16】 编写程序遍历数组每个元素，并输出。

程序源代码如下。

```

//Jpro2_16.java
1  public class Jpro2_16 {
2      public static void main(String[] args) {
3          int[] ns = { 1, 4, 9, 16, 25 };
4          for (int i=0; i<ns.length; i++) {
5              int n = ns[i];
6              System.out.println(n);
7          }
8      }
9  }

```

程序分析：第 5~8 行为数组遍历。为了实现 for 循环遍历，初始条件为 $i=0$ ，因为索引总是从 0 开始，继续循环的条件为 $i<ns.length$ ，因为当 $i=ns.length$ 时， i 已经超出了索引范围（索引范围是 $0 \sim ns.length-1$ ），每次循环后， $i++$ 。

如果直接输出每个元素，如何修改代码？



视频

2.4.4 多维数组

前文已介绍如何使用一维数组表示线性集合。当要处理 n 维的数据结构时，需要使用 n 维数组。例如，当表示一个矩阵或一个表格时，要使用二维数组。把 n 维数组称为多维数组。大多数情况下，程序中会使用多维数组中的二维数组，因此重点介绍二维数组。

1. 二维数组的声明、创建和初始化

声明二维数组的语法格式如下：

```
数据类型[][] 数组名;
```

数组名后的第一个方括号[]中的整数值是行下标，第二个方括号[]中的整数是列下标。与一

维数组一样，二维数组的每个下标是 `int` 型且从 0 开始。

例如，用上面格式声明整型的二维数组变量 `array`。

```
int[][] array;
```

创建二维数组的方法，使用关键字 `new`。格式如下：

```
数组名=new 数据类型[行长度][列长度];
```

例如

```
array=new int[5][5];
```

二维数组也可以以如下方式声明和创建。

```
数据类型[][] 数组名=new 数据类型[行数][列数];
```

通过给每个数组元素单个赋值，可以实现二维数组的初始化，例如：

```
array[0][0]=1;
array[2][4]=7;
```

也可以使用一个简化的方式来声明、创建和初始化二维数组。例如

```
int[][] array={{1,2,3},{4,5,6},{7,8,9},{11,12,13}};
```

2. 不规则数组

在二维数组中，每一行本身就是一个一维数组。因此这些行可以有不同的长度。这种数组称为不规则数组。例如

```
int[][] array={{1,2,3,4},{1,2,3},{1,2}};
```

则 `array[0].length=4`，`array[1].length=3`，`array[2].length=2`。

若预先不确定一个不规则数组中的列数，可以使用下面的语句创建。

```
int[][] array=new int[3][];
array[0]=new int[4];
array[1]=new int[3];
array[2]=new int[2];
```

注意：在这种创建方式中，`int[3][]`中的第一个下标不可省略。

【例 2.17】 编写程序，定义一个整型的 4 行 4 列的二维数组，并给数组第 0 行赋值 1、2、3、4，第 1 行赋值 5、6、7、8，以此类推。分行输出每行数组元素并求所有元素之和。

分析：定义二维数组 `b[4][4]`，各个元素值的通项公式为 `b[i][j]=i*4+j` (`i` 表示行，`j` 表示列)，用嵌套的双层循环实现程序功能。外层 `for` 语句控制数组 `b` 的各行，行下标从 0 开始到 `b.length-1`，内层 `for` 语句控制数组 `b` 的第 `i` 行的各列，列下标从 0 开始到 `b[i].length-1`。

程序源代码如下。

```
//文件名 Jpro2_17.java
1 public class Jpro2_17 {
2     public static void main(String[] args) {
3         int sum=0;
4         int[][] b=new int[4][4];           /*赋值
5         for(int i=0;i<b.length;i++)
6             for(int j=0;j<b[i].length;j++)
7                 b[i][j]=i*4+j;
8         System.out.print("数组元素如下：\n");
```

```

9      for(int i=0;i<b.length;i++){
10         for(int j=0;j<b[i].length;j++){
11             System.out.print("    "+b[i][j]);
12             sum+=b[i][j];                //求和
13         }
14         System.out.println();
15     }
16     System.out.println("二维数组所有元素的和等于"+sum);
17 }
18 }

```

程序运行结果如下:

数组元素如下:

```

0   1   2   3
4   5   6   7
8   9   10  11
12  13  14  15

```

二维数组所有元素的和等于 120

【练习2.4】

- [单选题]下面正确声明整型数组的语句是_____。

A. int Array[] a1,a2;	B. int[] a1,a2;
int a[]={1,2,3,4,5};	int a3[]={1,2,3,4,5};
C. int a1,a2[];	D. int[] a1,a2;
int a3={1,2,3,4,5};	int a3=(1,2,3,4,5);
- [单选题]若已定义:“int[] a=new int[10];”,则对 a 数组元素正确引用的是_____。

A. a[-3]	B. a[10]	C. a[5]	D. a(0)
----------	----------	---------	---------
- [单选题]定义语句“int a[]={11,22,33};”,则以下语句叙述错误的是_____。

A. 定义了一个名为 a 的一维数组	B. a 数组有三个元素
C. a 数组的下标为 1, 2, 3	D. 数组中的每个元素均为整型

2.5 实例

在无序的成绩表中寻找最高分和最低分是困难的,进行成绩分析时也较难比较,但是如果成绩是已按照由低到高排序的,就会比较方便对成绩做出分析。下面使用冒泡排序法对成绩进行排序。

【例 2.18】 某学期结束后,需对计算机专业学生的 Java 程序设计课程考试成绩进行排序。设计一个程序,从键盘输入学生的考试成绩,并按从低分到高分的顺序排序。

```

//文件名 Jpro2_18.java
1  import java.util.Scanner;
2  import java.util.Arrays;
3  public class Jpro2_18 {
4      public static void main(String[] args) {
5          Scanner scanner = new Scanner(System.in);
6          int[] score = new int[100];
7          int k = 0;
8          System.out.println("请输入成绩,输入-1 结束:");

```

```

9      while (true) {
10         int input = scanner.nextInt();
11         if (input == -1) break;
12         score[k] = input;
13         k++;
14     }
15     for (int i = 0; i < k - 1; i++) {
16         for (int j = 0; j < k - i - 1; j++) {
17             if (score[j] > score[j+1]) {
18                 int tmp = score[j];
19                 score[j] = score[j+1];
20                 score[j+1] = tmp;
21             }
22         }
23     }
24     System.out.print("排序后成绩: ");
25     for (int i = 0; i < k; i++) {
26         System.out.print(score[i] + " ");
27     }
28     scanner.close();
29 }
30 }

```

请读者自行运行程序并分析结果。

习题 2



1. 程序阅读题

(1) 分析下列程序，写出程序运行结果。

```

public class A{
    public static void main(String[] args) {
        int a=0;
        Label:for(;a<4;a++){
            for(int j=0;j<2;j++){
                if(a==2) break Label;
                System.out.print(a*2+j+"\t");
            }
            System.out.println("a="+a);
        }
    }
}

```

(2) 分析下列程序，写出程序运行结果。

```

public class A {
    public static void main(String[] args) {
        int a=8;
        if(a<8||a>8)
            System.out.println("a<>8");
        else
            System.out.println("a+8");
    }
}

```

(3) 分析下列程序，写出程序运行结果。

```
public class A{
    public static void main(String[] args) {
        int j=0;
        for(int i=3;i>0;i--) {
            j+=i;
            int x=2;
            while(x<i){
                x+=1;
                System.out.print(x);
            }
        }
    }
}
```

(4) 分析下列程序，写出程序运行结果。

```
public class A{
    public static void main(String args[]) {
        String str ="Hello,";
        str=str+"guys!";
        System.out.println(str);
    }
}
```

(5) 分析下列程序，写出程序运行结果。

```
public class A {
    public static void main(String[] args){
        int index=1;
        Boolean[] test=new Boolean[3];
        Boolean bb=test[index];
        System.out.println(bb);
    }
}
```

(6) 分析下列程序，写出程序运行结果。

```
public class A{
    public static void main(String[] args){
        int i;
        int a[]=new int[10];
        for(i=0;i<a.length;i++){
            a[i]=i*6+i;
        }
        for(i=1;i<a.length;i++){
            if(a[i]%3==0)
                System.out.println(a[i]);
        }
    }
}
```

2. 编程题

(1) 编写并测试一个Java类，计算一个一般三角形的面积。公式为： $\text{area} = \sqrt{p(p-a)(p-b)(p-c)}$ ，其中已知三角形三边 a 、 b 、 c ， $p=(a+b+c)/2$ 。

(2) 求 50 到 100 (包含 50 和 100) 之间的素数并输出。素数是指除 1 和它本身没有其他因子的自然数。

(3) 打印输出杨辉三角的前 10 行。

(4) 在数组中存放 10 个随机产生的整数, 输出数组, 并查找该数组中的最小数。

(5) 请用选择排序法将数列 2, 8, 4, 3, 7, 1, 5 按从小到大顺序排序, 选择排序就是每次寻找这一序列中最大的数, 并将其与序列的最后面数交换位置, 然后寻找未排序序列中的最大数与最后一个数交换位置, 以此类推, 直到全部排序完成。