《Android 系统下 Java 编程详解》

作者：华清远见

第 5 章 数组

本章简介

本章介绍了数组的基本概念；详细说明了一维数组的声明、创建和初始化方法；一维数组的内存空间和数据复制；数据结构的基础知识和如何使用一维数组实现数据结构和排序算法；最后简单介绍了多维数组的相关知识。

5.1 数组基本概念

数组是编程语言中非常常见的一种数据结构，用于储存一组有序数据的集合。数组中的每一个元素都属于同一数据类型，用一个统一的数组名和下标唯一地确定数组中的元素。可以通过整型索引访问数组中的每一个值。

可以通过数组来保存任何相同数据类型的数据：简单类型或者引用类型。数组本身属于引用类型。

数组被创建以后，它的大小是不能被改变的，但是，数组中的各个数组元素的值是可以被改变的。

5.2 一维数组

5.2.1 知识准备：一维数组的声明

定义一个一维数组很简单，可以使用下面两种方式：

```
数据类型[] 数组名
```

或

```
数据类型  数组名[]
```

通过上面两种方式中的一种，仅仅声明了一个数组变量，并没有创建一个真正的数组，也无法确定数组长度。这时候数组还不能被访问。上面两种声明数组的方式，可以任选一种，它们之间并没有优劣之分。一般来说，选择第一种方式比第二种方式要直观一些。

下面这段代码声明了不同数据类型的数组：

```
//int 数组
int[] intArray;
//字符型数组
char[] charArray;
//布尔型数组
boolean[] booleanArray;
//引用类型数组(字符串数组)
String[] stringArray;
//对象数组
MSG[] MsgArray;
```

5.2.2 知识准备：一维数组的创建

在声明了数组之后，就要具体规定数组的大小，给数组分配内存空间。可以通过 new 操作符来显示创建一个数组：

```
type[] arr_name;
arr_name = new type[length];
```

其中 type 是数组元素类型；arr_name 为自定义的数组名，命名规则和变量名相同，遵循标识符定名规则；length 是一个常量表达式，表示数组元素个数，即该数组的长度。

注意：

数组名后是用方括弧[]括起来的常量表达式，不能用圆括弧()，如 inta(10)；这样的用法是错误的。

数组创建示例：

```
int[] a;
a = new int[10];
```

这条语句创建了一个可以存储 10 个整型数据的数组，也就是分配了 10 个可以被 int 类型数据占用的内存空间。注意数组的索引从 0 开始，本例为 a[0]~a[9]。通过数组名和数组索引是访问数组元素的唯一方式，例如，要访问数组 a 的第一个元素，可以通过 a[0]来访问，要访问第 10 个元素，可以通过 a[9]来访问，要想访问第 n 个元素，可以通过 a[n-1]来访问。

为了方便操作数组，Java 语法中提供了获得数组长度的语法格式。对于一个已经创建完成的数组，获得该数组长度的语法格式为：“数组名.length”。例如上面的例子，可以用 a.length 来获得它的长度为 10。

当然，也可以更简便地将数组的声明和数组大小的分配放到一起来完成，如下：

```
type[] arr_name = new type[length];
```

例如，上面的数组 a 的声明和数组大小定义可以合并为如下的语句来完成：

```
int[] a = new int[10];
```

如果需要声明一个存放引用类型数据的数组，使用的方法也是一样的：

```
String[] s = new String[50];
```

5.2.3 任务一：一维数组的声明与创建实例

1. 任务描述

声明并创建一个数组用于存放开机密码，根据数组长度提示用户需要输入的密码位数。

2. 技能要点

- 使用 new 操作符创建数组。
- 获取数组长度。

3. 任务实现过程

（1）首先声明了一个 int 类型的数组 numberArray，然后，通过 new 操作符来给这个数组设置了它的长度，以及给这个数组分配了存放 6 个 int 类型数据的内存空间。通过 numberArray.length 可以得到这个数组的长度，在这里为 6。

源文件：NumberLength.java

```
public class NumberLength {

 public static void main(String[] args) {
      int[] numberArray;
      numberArray = new int[6];
      System.out.println("请输入长度为" + numberArray.length+"位开机密码");
 }
}
```

（2）编译并运行这个程序，将在控制台上得到如下的输出结果：

```
请输入长度为 6 位开机密码
```

5.2.4 知识准备：一维数组的初始化

Java 数组的初始化主要分为两种：静态初始化和动态初始化。在了解这两种初始化方式之前，先看一下 Java 提供的数组默认初始化。

Java 为了保证安全性，防止内存缺失，为已创建的数组提供了默认初始化机制。在创建成功一个数组后，将完成如下 3 个动作：

- 创建一个数组对象。
- 在内存中给数组分配存储空间。

- 给数组的元素初始化一个相应的数据类型的默认值。比如，将 int 类型的数组各个元素初始化为 0，引用类型是 null 等。

将任务一中的程序稍做修改，让它打印出数组第一个元素的默认值：

源文件：NumberLength.java

```
public class NumberLength {

 public static void main(String[] args) {
      int[] numberArray;
      numberArray = new int[6];
      System.out.println("请输入长度为" + numberArray.length+"位开机密码");
      System.out.println("第一位密码默认初始化值是：" + numberArray[0]);
 }
}
```

在这个程序中，首先声明了一个 int 类型的数组，然后，利用 new 操作符创建了一个长度为 6 的数组，它将给这个数组分配存储空间并且初始化这些数组元素，在这里将给这些数组元素一个值 0；最后，试图向控制台打印出这个数组第一个元素的值，因为 Java 中的数组索引（下标）是从 0 开始的，所以，第一个数组元素对应的索引为 0，所以可以通过 numberArray [0]的方式来得到数组的第一个元素的值。

编译并运行这个程序，将打印出如下的信息：

```
请输入长度为 6 位开机密码
第一位密码默认初始化值是：0
```

其实，此时在这个数组中的任何一个元素的值都是 0。读者可以自己修改数组的索引来获得不同的数组元素，注意这个数组的索引取值在 0~10 之间。

但是，通常情况下，定义一个数组并不是想用系统自动给的默认值，而是自己给数组的值。这时，就需要对数组进行初始化操作。也就是说，给数组的各个元素指定对象的值。

下面来看静态和动态这两种初始化方式的方法及它们的区别。

1）静态初始化

所谓静态初始化，就是在定义数组的时候就对数组进行初始化，如下：

```
int k[] = {1,3,5,7,9};
```

在这个例子中，定义了一个 int 类型的数组 k，并且用大括号中的数据对这个数组进行了初始化，各个数据之间用“,”分割开。此时数组的大小由大括号中的用于初始化数组的元素个数决定，注意不要在数组声明中指定数组的大小，否则将会引起错误。在这个例子中，将数组声明、数组的创建及数组的初始化都放在了同一条语句中，在这边并没有使用到 new 来创建这个数组。其等同于

```
int k[] =new int[5]
 k[] = {1,3,5,7,9};
```

来看一个静态初始化的例子：

源文件：StaticOpenNunber.java

```
public class StaticOpenNunber {

 public static void main(String[] args) {
      int numberArray[] = { 6, 5, 4, 3, 2, 1 };
      System.out.println("默认开机密码为");
      for (int i = 0; i < numberArray.length; i++) {
           System.out.println(numberArray[i]);
      }
 }
}
```

在这个例子中，利用静态方式对数组进行初始化。这个数组的长度是数组中的元素的个数：6。然后用一个 for 循环将数组的各个元素取出来打印到控制台。程序执行的结果如下：

```
默认开机密码为 654321
```

注意在这边用到了前面所论述的用于获得数组长度的方法：使用数组的 length 属性，用它来获得数组的长度。

在 Java 中，还可以利用静态初始化的方法来初始化一个匿名的数组，方法如下：

```
new type[] {… }
```

例如：

```
new String[] {"abc","cde","efg"}
```

可以通过这种方法来重新初始化一个数组，例如，有一个 String 类型的数组，它通过下面的静态方式被初始化：

```
String[] s = {"tom","jerry","mickey"};
```

此时，可以对 s 这个数组变量进行重新初始化，如下：

```
s = new String[]{"abc","cde","efg"};
```

这条语句等同于下面的两条语句：

```
String[] temp = {"abc","cde","efg"};
s = temp;
```

可以只给一部分元素赋值。例如：

```
int a[10]={0,1,2,3,4};
```

只给前 5 个元素赋值，后 5 个元素为 0。

初始化之后：a[0]=0，a[1]=1，a[2]=2，a[3]=3，a[4]=4，a[5]=0，…，a[8]=0，a[9]=0。

对全部数组元素赋初值时，可以不指定数组长度。例如：

```
int a[]={1,2,3,4,5};
```

上面的写法中，{ }中只有 5 个数，系统会据此自动定义数组的长度为 5。初始化之后：a[0]=1,a[1]=2,a[2]=3,a[4]=4,a[5]=5。如果被定义的数组长度与提供初值的个数不同，则数组长度不能省略。例如：

想定义数组长度为 10，就不能省略数组长度的定义。而必须写成：

```
int a[10]={ 1，2，3，4，5};
```

只初始化前面 5 个元素，后 5 个元素为 0。不能写成：

```
int a[ ]={1，2，3，4，5};
```

2）动态初始化

所谓动态初始化，就是将数组的定义和空间分配与给数组元素赋值分开。创建时系统进行数组的默认初始化。例如，对数组中的元素一个个地分别指定它们各自对应的值，这些赋值可以在程序的任意位置：

```
char ch = new char[3];
ch[0] = a;
ch[1] = b;
ch[2] = c;
```

或者用一个循环来对一个数组一次赋值，例如：

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

5.2.5 知识准备：引用数组元素

定义并用运算符 new 为之分配空间后，才可以引用数组中的每个元素，数组元素的引用方式为：

```
数组名[数组索引]
```

数组索引可以是一个整数或者一个整数表达式。依然要强调注意数组的索引从 0 开始到数组长度减 1。比如，数组长度为 n，则索引的范围为 0~(n-1)。

在使用数组名加数组索引的方式来取得数组的元素时，注意元素的索引必须小于数组的长度，也就是只能在 0~(n-1)之间，否则会引起数组越界的异常： java.lang.ArrayIndexOutOfBoundsException（关于异常将在后面章节中学习）。可以使用数组的一个属性 length 来获得数组的长度。

问：数组类型的变量也可以像基本类型那样整体引用吗？

答：这点正要提醒大家注意!

Java 语言规定只能逐个引用数组元素，不能一次引用整个数组。

5.2.6 任务二：引用数组实例，对数组排序

1. 任务描述

声明并初始化一个字符类型数组，对数组元素进行由小到大的排序，并输出数组元素。

2. 技能要点

- 掌握数组的静态和动态初始化。
- 掌握根据索引号引用数组元素。

3. 任务实现过程

对于数组的排序，可以像前面的一维数组应用的例子一样，自己写一个算法来对数组进行排序。但是，也可以利用 java.util 包中的 Arrays 类的一个静态方法 sort 来进行数组元素的排序。这个方法有一个数组类型参数，用来接收需要进行排序的数组。

源文件：ArraysSort.java

```
import java.util.Arrays;
public class ArraySort {
 public static void main(String[] args) {
      char[] a = { 'd', 'c', 'b', 'a', 'f', 'e'};
      System.out.println("Before Sorting:");
      for (int i = 0; i < a.length; i++) {
           System.out.print("a[" + i + "]=" + a[i] + "    ");
      }
      System.out.println("");
      Arrays.sort(a);
      System.out.println("After Sorting:");
      for (int i = 0; i < a.length; i++) {
           System.out.print("a[" + i + "]=" + a[i] + "    ");
      }
 }
}
```

运行这个程序后，在控制台上将打印出如下的信息：

```
Before Sorting:
a[0]=d    a[1]=c    a[2]=b    a[3]=a    a[4]=f    a[5]=e
After Sorting:
a[0]=a    a[1]=b    a[2]=c    a[3]=d    a[4]=e    a[5]=f
```

可以看出，通过 Arrays 的静态方法 sort()，已经将数组 a 中的所有元素按照从小到大排好了顺序。

5.2.7 知识准备：简单数据类型数组的内存空间

通过以上的学习，已经知道了数组初始化的一些步骤，下面来了解一下数组的内存空间和内存分配。一般 Java 在内存分配时会涉及以下区域。

- 寄存器：在程序中是无法控制的。
- 栈：存放基本类型和对象的引用，但对象本身不存放在栈中，而是存放在堆中。
- 堆：存放用 new 产生的数据。

- 静态域：存放在对象中用 static 定义的静态成员。
- 常量池：存放常量。
- 非 RAM 存储：硬盘等永久存储空间。

首先来看简单数据类型数组从定义到初始化的内存变化过程。

1．简单数据类型数组的声明

在声明数组的时候，系统会给这个数组分配用于存放这个数组的内存空间：它会在堆（Heap）内存空间中给数组分配一个空间用于存放数组引用变量。在栈内分配空间存入数组对象的引用，如图 5-1 所示。

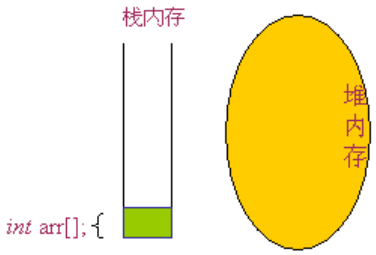

图 5-1　int 类型数组的定义

2．简单数据类型数组的创建

在创建简单数据类型的数组的时候，系统会分配合适的堆空间用来存放该种数据类型数据的内存空间，并且将这个数组的各个元素赋一个和数组类型匹配的初值。

具体到这个 int 类型数组的例子，所有的数组元素都会被初始化成 0，如图 5-2 所示。

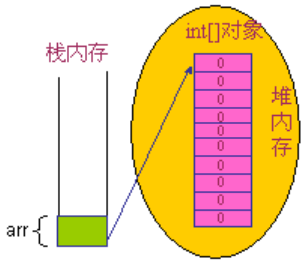

图 5-2　int 类型数组的创建

3．简单数据类型数组的初始化

当对数组进行初始化时，会将值赋给对应的各个数组元素。比如，通过下面的一个循环对这个 int 类型的数组进行初始化：

```
for(int i=0;i<10;i++){
    arr[i]=i+1;
}
```

则会将 1~10 的值赋给这个长度为 10 的 int 类型数组，如图 5-3 所示。

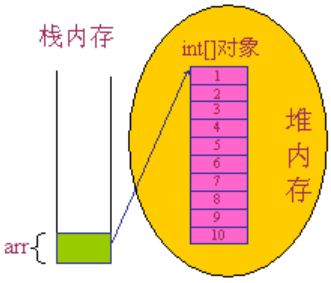

图 5-3　int 类型数组的初始化

4．引用数据类型数组的内存空间

在介绍完简单数据类型数组的初始化过程后，再来看引用类型数组的初始化过程中的内存变化。

5．引用数据类型数组的声明

引用类型数组的定义和简单数据类型数组的定义基本相同。

图 5-4 所示为执行下面操作后的结果：

```
String[] arr;
```

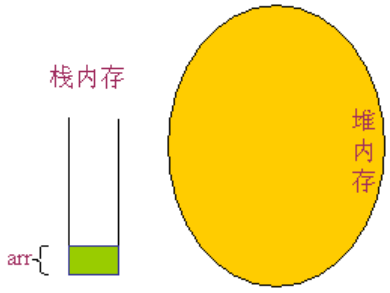

图 5-4　引用数据类型数组的定义

6．引用数据类型数组的创建

引用数据类型数组在创建的时候也是首先给数组元素分配内存空间，然后赋给这些数组元素一个默认的初始值 null。

图 5-5 所示为执行下面操作后的结果：

```
arr = new String[10];
```

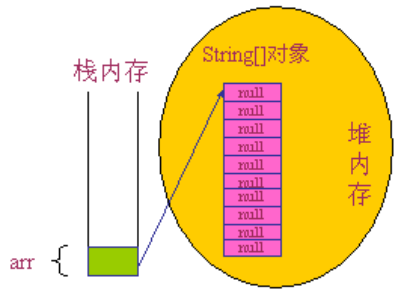

图 5-5　引用数据类型数组的创建

7．引用数据类型数组的初始化

在进行引用数据类型数组的初始化的时候，和简单数据类型数组的初始化有些不同，因为数组本身是引用类型，而现在数组元素也是引用类型，所以这个时候需要给数组元素所引用的对象也分配内存空间。

图 5-6 所示为执行下列操作的结果：

```
arr[0]=new String("one");
arr[1]=new String("two");
arr[2]=new String("three");
arr[3]=new String("four");
arr[4]=new String("five");
arr[5]=new String("six");
arr[6]=new String("seven");
arr[7]=new String("eight");
arr[8]=new String("night");
arr[9]=new String("ten");
```

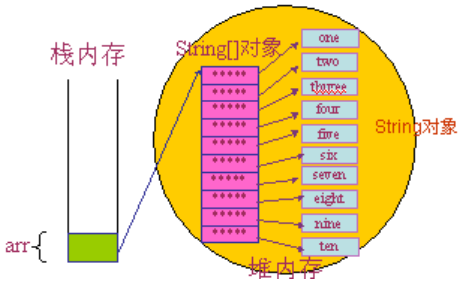

图 5-6　引用类型数组的初始化

5.2.8　技能拓展任务：数组复制

1．任务描述

声明一个数组变量存放一串电话号码，将此数组变量复制给另一个数组变量以做备份。备份后修改任何一个号码，另一个号码也同时被修改。

2．技能要点

- 了解数组对象创建后在内存中的存放机制。
- 编写算法进行数据复制。

3．任务实现过程

要求体现出复制之后的数组变量和原数组变量是对同一数组对象的引用。

这两个数组变量均指向同一个数组；通过任何一个数组变量进行操作，均会对另一个数组变量中的数组产生影响。

可以将一个数组变量复制给另一个数组变量，这个时候，这两个数组变量均指向同一个数组。通过任何一个数组变量进行操作，均会对另一个数组变量中的数组产生影响。

请看下面这个例子：

源文件：NumberBackup.java

```
public class NumberBackup {
 public static void main(String[] args) {
      int[] a = { 6, 7, 3, 4, 1, 9, 0, 5 };
      int[] b;
      b = a;
      System.out.println("Before Backup:");
      for (int i = 0; i < a.length; i++) {
           System.out.print("a[" + i + "]=" + a[i] + "    ");
      }
      System.out.println("\n"+"Backup a to b");
      for (int i = 0; i < b.length; i++) {
           System.out.print("b[" + i + "]=" + b[i] + "    ");
      }
      b[3] = 8;
      System.out.println("\n"+"After Modifying b :");
      for (int i = 0; i < b.length; i++) {
           System.out.print("b[" + i + "]=" + b[i] + "    ");
      }
      System.out.println("\n");
      for (int i = 0; i < a.length; i++) {
           System.out.print("a[" + i + "]=" + a[i] + "    ");
      }
 }
}
```

这个程序首先初始化了一个 int 类型的数组 a，然后，将这个数组变量复制给另一个 int 类型的数组 b，这个时候，数组变量 a 和 b 均指向同一个数组，如果通过数组变量 b 来对数组的内容进行修改，也会反映到数组变量 a 中。

执行这个程序后的结果是：

```
Before Backup:
a[0]=6    a[1]=7    a[2]=3    a[3]=4    a[4]=1    a[5]=9
Backup a to b
b[0]=6    b[1]=7    b[2]=3    b[3]=4    b[4]=1    b[5]=9
After Modifying b :
b[0]=6    b[1]=7    b[2]=3    b[3]=8    b[4]=1    b[5]=9
a[0]=6    a[1]=7    a[2]=3    a[3]=8    a[4]=1    a[5]=9
```

注意看 a[3]数组元素的值的变化。图 5-7 揭示了以上程序的执行过程。

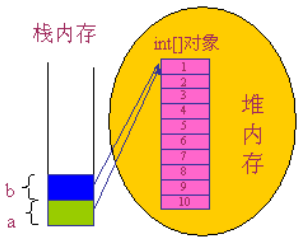

图 5-7　通过数组变量实现的数组复制

5.3　数据结构及数组应用

5.3.1　知识准备：堆栈

堆栈（Stack），也称为“栈”，是一种简单的、使用广泛的数据结构。堆栈是一种特殊的序列，这种序列只在其中的一头进行数据的插入和删除操作，通常将这头称为“栈顶”。相应地，另一头称为栈底。不含任何数据（元素）的堆栈称为空栈。图 5-8 是堆栈的示意图。

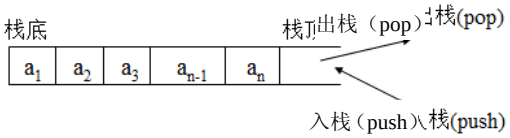

图 5-8　堆栈示意图

在这幅示意图中，可以看出堆栈的一些特点：栈可以看做先进后出（FILO）的线性表。或后进先出（LIFO）的线性表，表尾进行插入和删除的操作。也就是说，在堆栈中，堆栈元素（数据）的操作都是遵循“后进先出（Last In First Out，LIFO）”的原则进行的。在实际开发中，只要问题满足“后进先出”原则，都可以使用堆栈来解决。

在 Java 中，如果要定义一个类来表示堆栈，通常需要让这个类能够实现下面的功能：

- 入栈：将元素往堆栈中添加，这个动作也经常被称做“压栈（Push）”。
- 出栈：将元素从堆栈中取出来，这个动作也经常被称做“弹栈（Pop）”。
- 清空：将堆栈中的所有元素都清空。
- 返回顶端元素：获得最顶端的元素，但不将它从堆栈中删除。

5.3.2　任务三：使用数组实现堆栈

1．任务描述

使用数组实现利用堆栈记录并查询手机通话记录。

2．技能要点

- 掌握堆栈先进后出的结构特点。
- 使用数组实现堆栈的入栈出栈操作。

3．任务实现过程

（1）编写一个类，在类中定义一个数组，用来记录手机通话。编写出栈入栈的方法来操作数组。

源文件：MyCall.java

```
class MyCall {
 private int capacity = 100;
 private String[] items;
 private int top = 0;
 // 不带参数构造器
 public MyCall() {
      this(100);
 }
 // 带参数构造器，参数为堆栈大小
 public MyCall(int cap) {
      this.capacity = cap;
      items = new String[cap];
 }
 // 入栈
 public void push(String s) {
      top++;
      items[top] = s;
 }
 // 出栈
 public void pop() {
      items[top] = null;
      top--;
 }
 // 清空堆栈
 public void empty() {
      top = 0;
 }
 // 取出最顶端的堆栈元素
 public String top() {
      return items[top];
 }
 // 获得堆栈元素个数
 public int size() {
      return top;
 }
}
```

（2）然后编写一个类，在这个类中，使用上一个类中堆栈的相关方法来操作堆栈。

源文件：CallHistory.java

```
public class CallHistory {

 public static void main(String[] args) {
      MyCall mc = new MyCall(5);
      mc.push("Mia 16:12");
      mc.push("Kathy 18:02");
      mc.push("Alex 19:35");
      System.out.println("I have "+mc.size()+" calls today");
      System.out.println("The latest one is: "+mc.top());
 }
}
```

（3）编译并运行这个程序，将得到如下的输出：

```
I have 3 calls today
The latest one is: Alex 19:35
```

可以看到，输出结果是将最后入栈的记录“Alex 19:35”最先弹出显示。

提示：

上面的 Stack 例子中，其实不是很完美，至少有两个地方是 Stack 没有考虑到的：当堆栈中的数据已经满的时候，如果再试图进行压栈操作，将会发生错误；如堆栈中没有数据，那么，如果此时试图进行弹栈操作，也将会发生错误。一个可行的实现方式是在压栈或者弹栈操作的时候，对栈内的空间进行判断，在发生上述问题的时候抛出一个 RuntimeException 类型的异常（应该自己定义一个 RuntimeException 子类）。关于异常，请参考后续章节的内容。

5.3.3 知识准备：队列

队列（Queue）是另外一种常用的数据结构。它也是一种特殊的线性表，对这种线性表，删除操作只在表头（称为队头）进行，插入操作只在表尾（称为队尾）进行。队列的修改是按先进先出的原则进行的，所以队列又称为先进先出（First In First Out）表，简称 FIFO 表。这种特点和队列的名字是很相符的，就像生活中的排队一样：排在队伍最前面的人最先得到相关的服务，也最先从队伍中出来。与堆栈中的数据操作总是在栈顶进行不同，在队列中，它的两头都能够进行操作，在队头（front）删除元素，而在队尾（rear）加入元素，如图 5-9 所示。

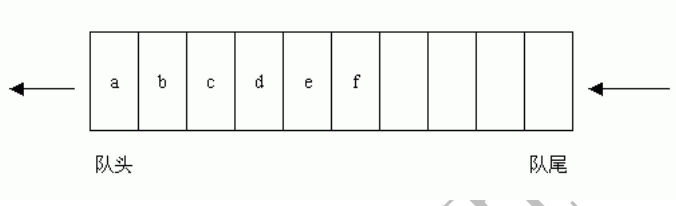

图 5-9　队列示意图

一个典型的队列可以实现以下功能。

- 往队列的队尾插入一个元素（enqueue）。
- 将队列的队头元素删除（dequeue）。
- 清空队列（makeEmpty）。
- 判断队列是否为空（isEmpty）。

5.3.4 任务四：使用数组实现队列

1．任务描述

使用数组实现利用队列记录并查询手机通话记录。

2．技能要点

- 掌握队列先进先出的结构特点。
- 使用数组实现队列入列、出列等基本操作。

3．任务实现过程

（1）编写一个类，在类中定义一个数组，用来记录手机通话。使用队列的数据结构操作数组，并编写入列、出列，判断队列是否为空，判断队列是否已满的操作。

源文件：CallHistory2.java

```
class MyQueue {
 // 队头和队尾索引
 private int front = -1, rear = -1;

 // 定义一个数组模拟队列
 private String[] queue;

 // 构造器，参数 maxElements 为队列长度
 public MyQueue(int maxElements) {
      queue = new String[maxElements];
 }
```

```
    // 入列
    public void enqueue(String e) {
         queue[++rear] = e;
    }

    // 判断队列是否为空
    public boolean isEmpty() {
         return front == rear;
    }

    // 判断队列是否已满
    public boolean isFull() {
         return rear == queue.length - 1;
    }

    // 出列
    public String dequeue() {
         return queue[++front];
    }
}
```

在这个类中，定义了一个数组用于模拟队列，并且定义了 enqueue()用于入列，dequeue()方法用于出列，另外，还有两个方法 isFull()和 isEmpty()分别用于判断队列是否已经满了或者队列是否为空。

（2）编写一个类，在这个类中，使用上一个类中队列的相关方法来操作队列。

源文件：CallHistory2.java

```
public class CallHistory2 {

     public static void main(String[] args) {
          MyQueue queue = new MyQueue(20);
          queue.enqueue("Mia 16:12");
          queue.enqueue("Kathy 18:02");
          queue.enqueue("Alex 19:35");
          System.out.println(queue.dequeue());
          System.out.println(queue.dequeue());
          System.out.println(queue.dequeue());
     }
}
```

编译并运行这个程序，可以得到如下的输出：

```
Mia 16:12
Kathy 18:02
Alex 19:35
```

可以看到，使用队列的数据结构是，先入列的记录首先输出，最后入列的记录最后输出。

5.3.5 知识准备：排序算法

在编写程序的时候，经常会碰到算法问题。所谓算法（Algorithm），就是在有限步骤内求解某一问题所使用的一组定义明确的规则。算法的好坏直接影响到程序的运行效率，因此，选择一个好的算法对于编写高效的程序是至关重要的。本节以编程中常用的几种排序算法为例，来初步讲解算法，并结合数组的相关知识，加深对数组应用的理解。

1．冒泡排序法

对几个无序的数字进行排序，最常用的方法是所谓的冒泡排序法。算法思想是每次比较两个相邻的数，将较小的放到前面，较大的放到后面，这样就可以将这些数中最大的找出来放到最后，然后比较剩下的数，再在这些数中找出最大的，直到所有的数字按照从小到大的顺序排列。

可以用一个一维数组来存放这些需要进行排序的数字，然后对这个一维数组使用上面的算法对它的数组元素进行冒泡排序。下面是一种实现方式。

源文件：BubbleSort.java

```
public class BubbleSort {
 static String sortArray(int before[]) {
      String result = "";
      for (int i = 0; i < before.length; i++) {
           result += before[i] + " ";
      }
      return result;
 }

 static int[] bubbleSort(int before[]) {
      int t;
      for (int i = 0; i < before.length; i++) {
           for (int j = 0; j < before.length - i - 1; j++) {
                if (before[j] > before[j + 1]) {
                     t = before[j];
                     before[j] = before[j + 1];
                     before[j + 1] = t;
                }
           }
      }
      return before;
 }

 public static void main(String args[]) {
      int a[] = { 12, 43, 23, 56, 8, 22, 65, 87 };
      System.out.println("Before sorting:" + sortArray(a));
      a = bubbleSort(a);
      System.out.println("After Sorting:" + sortArray(a));
 }
}
```

通过类 BubbleSort 中的 bubbleSort()方法，可以对数组元素实现冒泡排序。并返回一个排好序的新的数组。

2．选择排序

选择排序的基本思想是：每一趟从待排序的数据元素中选出最小（或最大）的一个元素，顺序放在已排好序的数列的最后，直到全部待排序的数据元素排完。选择排序相对于冒泡来说，它不是每次发现逆序都交换。选择排序是不稳定的排序方法。

假设有一个数组如下：

```
初始关键字 [49 38 65 97 76 13 27 49]
第一趟排序后 13 [38 65 97 76 49 27 49]
第二趟排序后 13 27 [65 97 76 49 38 49]
第三趟排序后 13 27 38 [97 76 49 65 49]
第四趟排序后 13 27 38 49 [76 97 65 49 ]
第五趟排序后 13 27 38 49 49 [97 65 76]
第六趟排序后 13 27 38 49 49 65 [97 76]
第七趟排序后 13 27 38 49 49 65 76 [97]
```

最后排序结果为 13 27 38 49 49 65 76 97，下面来看一个使用 Java 实现快速排序算法的例子：

源文件：SelectionSort.java

```
public class SelectionSort {
 // 选择排序方法
 public static void selectionSort(int[] number) {
      for (int i = 0; i < number.length - 1; i++) {
           int m = i;
           for (int j = i + 1; j < number.length; j++) {
                if (number[j] < number[m])
```

```
                    m = j;
               }
               if (i != m)
                    swap(number, i, m);
          }
     }

     // 用于交换数组中的索引为 i、j 的元素
     private static void swap(int[] number, int i, int j) {
          int t;
          t = number[i];
          number[i] = number[j];
          number[j] = t;
     }

     public static void main(String[] args) {
          // 定义一个数组
          int[] num = { 2, 1, 5, 876, 12, 56 };
          // 排序
          selectionSort(num);
          for (int i = 0; i < num.length; i++) {
               System.out.println(num[i]);
          }
     }
}
```

3．快速排序

快速排序是对冒泡排序算法的改进，是目前使用最广泛的排序算法。快速排序的基本思路是：将一个大数组的排序问题，分解成两个小的数组的排序。而每一个小的数组又可以继续分解成更小的两个数组，这样这个数组的排序方式可以一直的递归分解下去，直到数组的大小最大为 2。在第一次划分的时候，选择一个基准元素，然后将它分成左右两个无序的数组，并且，使得左边的所有数组元素都小于等于基准元素，而右边的所有数组元素都大于等于基准元素，然后分别对左边和右边的数组递归做同样的操作。在这个排序方法中，算法效率的关键在于如何分解数组，也就是如何确定数组的基准元素。通常情况下可以选择数组最左边的元素作为基准元素，或者选择数组中间的元素作为基准元素。下面来看一个例子：

源文件：QuickSort.java

```
public class QuickSort {
     // 排序方法，接受一个 int[]参数，将会调用快速排序方法进行排序
     public static void sort(int[] number) {
          quickSort(number, 0, number.length - 1);
     }

     // 快速排序方法
     private static void quickSort(int[] number, int left, int right) {
          if (left < right) {
               int s = number[left];
               int i = left;
               int j = right + 1;
               while (true) {
                    // 向右找大于 s 的数的索引
                    while (i + 1 < number.length && number[++i] < s)
                         ;
                    // 向左找小于 s 的数的索引
                    while (j - 1 > -1 && number[--j] > s)
                         ;
                    // 如果 i>=j，退出循环
                    if (i >= j)
                         break;
                    // 否则交换索引 i 和 j 的元素
                    swap(number, i, j);
               }
               number[left] = number[j];
```

```
            number[j] = s;
            // 对左边进行递归
            quickSort(number, left, j - 1);
            // 对右边进行递归
            quickSort(number, j + 1, right);
        }
    }

    // 交换数组 number 中的索引为 i、j 的元素
    private static void swap(int[] number, int i, int j) {
        int t;
        t = number[i];
        number[i] = number[j];
        number[j] = t;
    }

    public static void main(String[] args) {
        int[] num = { 34, 1, 23, 345, 12, 546, 131, 54, 78, 6543, 321, 85,
                1234, 7, 76, 234 };
        sort(num);
        for (int i = 0; i < num.length; i++) {
            System.out.println(num[i]);
        }
    }
}
```

提示：

快速排序采用了分治法的基本思想，它将原问题分解为若干个规模更小但结构与原问题相似的子问题。递归地解这些子问题，然后将这些子问题的解组合为原问题的解。

5.3.6 技能拓展任务：排序算法实例

1. 任务描述

使用上述一种排序算法，在一个数组中存放近十天内手机上网流量使用情况，并按从大到小的顺序输出。

2. 技能要点

- 掌握使用数组实现各种排序算法。
- 灵活运用各种排序算法解决实际问题。

3. 任务实现过程

（1）使用选择排序算法。

```
public class GPRSFlow {

    // 选择排序方法
    public static void flowSort(double[] number) {
        for (int i = 0; i < number.length - 1; i++) {
            int m = i;
            for (int j = i + 1; j < number.length; j++) {
                if (number[j] > number[m])
                    m = j;
            }
            if (i != m)
                swap(number, i, m);
        }
    }

    // 用于交换数组中的索引为 i、j 的元素
    private static void swap(double[] number, int i, int j) {
        double t;
```

```
        t = number[i];
        number[i] = number[j];
        number[j] = t;
    }
    public static void main(String[] args) {
        // 定义一个数组
        double[] num = { 5.2,7.0 , 2.1, 0.9, 12.8, 13.0,3.4,4.4,10.0,15.0 };
        // 排序
        flowSort(num);
        for (int i = 0; i < num.length; i++) {
            System.out.println(num[i]);
        }
    }
}
```

（2）运行程序，得到如下输出：

```
15.0
13.0
12.8
10.0
7.0
5.2
4.4
3.4
2.1
0.9
```

可以看到，只是将上一节中的选择排序算法进行了小幅修改，改变了数据类型和输出顺序。所以上面几个排序算法应该熟记。

5.4 多维数组

Java 中支持多维数组，也就是“数组的数组”。多维数组每一维（最里面的一维不算），本身是一个一般数组，只不过里面的每一个元素的类型，是一个维度比它小一维的另一个数组。

5.4.1 知识准备：多维数组的声明

多维数组的声明是通过每一维一组方括号的方式来实现的。

格式：类型说明符 数组名[常量表达式 1] [常量表达式 2]

二维数组：int[][],double[][]等。

三维数组：float[][][],String[][][]等。

可以用数学的思维来理解多维数组，如二维数组中元素的排列顺序是：先行后列。 因此，可以把二维数组看成是一个矩阵。

5.4.2 知识准备：多维数组的创建

当使用 new 来创建多维数组时，不必指定每一维的大小，而只需指定最左边的维的大小就可以。如果指定了其中的某一维的大小，那么所有处于这一维左边的各维的大小都需要指定。

下面是一些创建多维数组的例子：

```
boolean[][] b = new boolean[10][3];
int[][] a = new int[5][];
String[][][] = new String[4][5][6]
double[][][] = new double[40][][]
```

下面的创建方式是错误的：

```
//int[ ][ ] a = new int [ ][5];
```

5.4.3 知识准备：多维数组的初始化

在知道数组元素的情况下，可以直接初始化数组，不必调用 new 来创建数组，这和一维数组的静态初始化类似：

分行给二维数组赋初值。

```
int a[2][3]={{1,2,3},{4,5,6}};
```

按数组的排列顺序对各数组元素赋初值。

```
int b[2][3]={1,2,3,4,5,6};
```

可以对部分元素赋初值。

```
int c[3][4]={{1},{5},{9}};
int d[3][4]={{1},{5,6},{0,9,7}};
```

在对全部数组元素赋初值时，数组第一维的长度可以不指定。

```
int e[ ][3]={1,2,3,4,5,6};
int f[ ][4]={{0,0,3},{0},{0,10}};
```

在引用多维数组的时候，通过指定数组名和各维的索引来引用。

注意：

二维数组元素仍然是从 a[0][0]开始。

除了静态初始化外，多维数组也可以通过数组声明和初始化分开的动态初始化方法来对数组进行初始化，例如：

```
int a[][] = new int[4][5];
int b[][] = new int[3][]
b[0] = new int[4];
b[1] = new int[3];
b[2] = new int[5];
```

5.5 本章小结

本章对一维和多维数组的基本概念，声明、创建和初始化方式进行了介绍。结合数组内存空间的分配的相关知识使同学们加深了解数组的原理和使用方式。着重介绍了队列，堆栈等数据结构、排序算法与数组的结合使用。通过学习本章，同学们应该掌握数组基本的使用方法。数组的使用很广泛也很灵活，同学们在今后的开发中要结合内存空间，程序效率等多方面合理使用数组。

课后练习题

一、选择题

1．编译并且执行以下代码结果是（　　）。

```
public class Test{
public static void main(String args[]){
int array[]=new int[]{3,2,1};
System.out.println(array[1]);
}
}
```

A．1　　B．2　　C．3　　D．有错误，数组的大小没有定义

2．编译并且执行以下代码结果会是（　　）。

```
 public class Test{
public static void main(String args[]){
String array[]=new String[5];
```

```
System.out.println(array[0]);
}
}
```

A．不确定的值　　B．0

C．null　　D．有错误，数组没有初始化

3．下面（　　）代码能够正确的计算出由命令行传递给应用程序的参数个数。

A．int count=args.length;

B．int count=args.length-1;

C．int count=0;
```
while(args[count]!=0)
  count++;
```

D．int count=0;
```
while(!(args[count].equals("")))
  count++;
```

4．能正确定义一个数组的代码是（　　）。

A．int [][] num=new int{{1},{2,3}};　　B．int num[][5];

C．int num[][]=new int[5][];　　D．int num[][5]=new int[][5];

5．阅读下面代码，运行结果是（　　）。

```
public class  Test{
public static void main(String[] args){
     int i;
     int f[]=new int[6];
     f[0]=f[1]=1;
     for(i=2;i<6;i++)
     {
          f[i]=f[i-1]+f[i-2];
     }
     for(i=0;i<6;i++)
     {
          System.out.print(f[i]+" ");
     }
}
}
```

A．1　2　3　5　8　　B．1　1　2　3　5　8

C．2　3　5　8　　D．1　1　2　3　5

二、填空题

数组用来存储一组_____数据。可以通过_____访问数组中的每一个值。可以通过数组来保存任何相同数据类型的数据：简单类型或者引用类型。数组本身属于____类型。数组被创建以后，它的_____是不能被改变的，但是，数组中的各个数组元素是可以被改变的。

三、编程题

1. 编写一个方法，方法的返回值为一个数组，数组里存放 Student 类的对象，数组的大小由参数传入。在程序中调用此方法，输出数组中各元素的值。

2．在一个数组中存放 10 个学生信息（姓名，成绩），然后按成绩从大到小排列输出。

联系方式

企业学院：www.farsight.com.cn　　物联网学院：www.topsight.cn　　研发中心：dev.hqyj.com

集团总部地址：北京市海淀区西三旗悦秀路北京明园大学校内 华清远见教育集团

北京地址：北京市海淀区西三旗悦秀路北京明园大学校区，电话：010-82600386/5

上海地址：上海市徐汇区漕溪路 250 号银海大厦 11 层 B 区，电话：021-54485127

深圳地址：深圳市龙华新区人民北路美丽 AAA 大厦 15 层，电话：0755-22193762

成都地址：成都市武侯区科华北路 99 号科华大厦 6 层，电话：028-85405115

南京地址：南京市白下区汉中路 185 号鸿运大厦 10 层，电话：025-86551900

武汉地址：武汉市工程大学卓刀泉校区科技孵化器大楼 8 层，电话：027-87804688

西安地址：西安市高新区高新一路 12 号创业大厦 D3 楼 5 层，电话：029-68785218