第 3 章　Visual Basic 数组

本章要点：

本章介绍 Visual Basic 程序设计中有关数组使用的基本语法知识，知识要点包括：

(1) 数组的概念；

(2) 数组(一、二维)的定义及其引用；

(3) 动态数组的含义及其使用方法；

(4) 与数组相关的函数的使用方法；

(5) 与数组有关的常用算法，如排序、插入等。

案例 3-1　输入 100 个学生成绩，并输出其平均值

【案例效果】

在本例中，设计程序完成一个输入 100 个学生的成绩并求其平均成绩的任务，程序运行后，单击“输入”按钮，弹出如图 3-1 所示成绩输入窗口，输入 100 个学生成绩后，界面打印输出其平均成绩，运行结果如图 3-2 所示。

图 3-1　成绩输入窗口

图 3-2　成绩运行结果

【设计过程】

1. 界面设计

启动 Visual Basic 6.0，新建一个“标准 EXE”工程，然后在窗体 Form1 上绘制一个命令按钮(Command1)。界面如图 3-3 所示。

2. 属性设置

在属性窗口里进行属性设置，如表 3-1 所示。

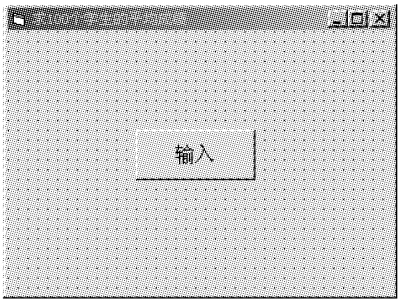

图 3-3　求 100 个学生的平均成绩界面设计

表 3-1　属性设置

对　象	属性名	属 性 值
Form1	Caption	"求 100 个学生的平均成绩"
Command1	Caption	* "输入"

3. 代码设计

在代码编辑器窗口中 Command1 的 Click 事件中编写如下代码：

```
Private Sub Command1_Click()
    Dim i As Integer,a(100) As Integer
    Dim s As Single,avg As Single
    s=0
    For i=1 To 100
      a(i)=Val(InputBox("请输入第"& i &"个学生成绩"))
      s=s+a(i)
    Next i
    avg=s/100
    Print"平均成绩为";avg
End sub
```

【相关知识】

1. 数组的概念

数组是一组具有相同类型和名称的变量的集合。如：A(1 To 100)，表示一个包含 100 个变量的名为 A 的数组。这些变量称为数组的元素，每个数组元素都有一个编号，这个编号叫做下标，我们可以通过下标来区别这些元素。数组元素的个数有时也称之为数组的长度。

数组元素中下标的个数称为数组的维数。例如案例 3-1 中，语句 Dim a(100) As Integer 定义了一个名为 a 的整型数组，来存放 100 个学生的成绩，数组 a(100)中只有一个下标，所以称为一维数组。

2. 数组的定义

数组必须先定义后使用。数组的定义就是让系统在内存中分配一个连续的区域，用

来存储数组元素。

(1) 一维数组的定义。静态一维数组的定义格式：

```
Dim 数组名(下标)[As 类型]
```

说明：在数组定义时，下标必须为常数，不可以为表达式或变量；下标下界最小为－32 768，最大上界为 32 767；省略下界，其默认值为 0，一维数组的大小为：上界－下界＋1。[As 类型]用于定义数组元素的数据类型，如果省略，则为变体型。

(2) 多维数组的定义

静态多维数组的定义格式：

```
Dim 数组名(下标 1[,下标 2,…])[As 类型]
```

说明：下标个数决定数组的维数，最多 60 维。每一维的大小＝上界－下界＋1；数组的大小＝每一维大小的乘积。例：

```
Dim C(-1 To 5,4)As Long
```

声明了 C 是数组名、长整型、二维数组、第一维下标范围为－1～5，第二维下标的范围是 0～4，占据 7×5 个长整型变量的空间。

(3) 注意事项

在有些语言中，下界一般从 1 开始，为了便于使用，在 Visual Basic 的窗体层或标准模块层，用 Option Base *n* 语句可重新设定数组的默认下界，如 Option Base 1。

在数组定义时的下标只能是常数，而在数组元素的使用时，数组元素的下标可以是变量或表达式。

3. 数组元素的基本操作

(1) 数组元素的引用。数组经定义后，才可在程序中使用。Visual Basic 中不能直接存取整个数组，对数组进行操作时需要引用数组的元素，使用格式如下：

```
数组(下标)
```

在引用数组元素时，数组名、数据类型和维数必须和定义时一致。另外还要注意区分数组的定义和数组元素的引用，例如下面的程序片段：

```
Dim x(8)As Integer
Dim Temp As Integer
…
Temp=x (8)
```

尽管有两个 x(8)，但是语句“Dim x (8) As Integer”中的 x(8)不是数组元素，而是说明由它声明的数组 x 的下标最大值为 8；而赋值语句

```
Temp=x(8)
```

中的 x(8)是一个数组元素。

(2) 数组的赋值与输入。在程序中,凡是简单变量出现的地方都可以用数组元素代替。普通变量赋值的方法同样适用于数组元素。例如:

```
a(1)=123
b(5)='www'
```

如果数组较大,用单个赋值语句逐个给数组元素赋值,就会使程序相当长,此时也可使用 Array 函数来为数组赋值。其使用方法将在后面的案例中进行介绍。

需要对数组中每个元素逐一赋值,可以运用循环语句。如案例 3-1 中运用 for 语句,使用 InputBox 函数来完成 100 个学生成绩的输入。代码如下:

```
For i=1 To 100
    a(i)=Val(InputBox("请输入第"& i &"个学生成绩"))
    s=s+a (i)
Next i
```

(3) 数组元素的输出。数组元素的输出可以使用循环语句和 Print 语句来实现,例如要输出 a 数组中的 100 个元素,可使用下面的语句:

```
For i=1 To 100
Print a(i);" ";
Next i
```

案例 3-2　二维数组使用举例

【案例效果】

设计程序,完成一个相同阶数的矩阵的加法。程序运行时首先启动如图 3-4 所示的窗体,单击窗体上的"矩阵求和"按钮,系统随机产生两个 3×4 的矩阵 ***A*** 和 ***B***,并把两个矩阵的加法结果显示在矩阵 ***C*** 中,运行效果如图 3-5 所示。

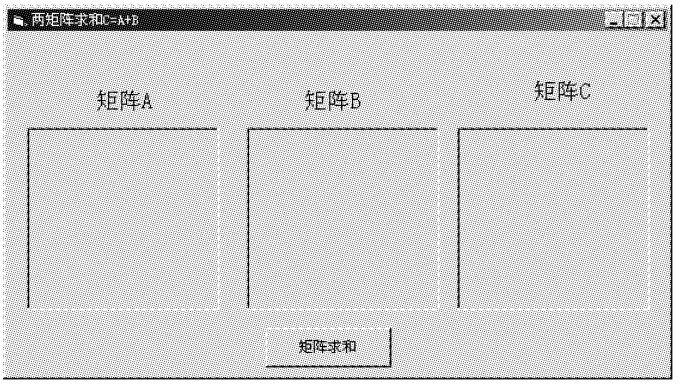

图 3-4　矩阵的加法运行初始界面图

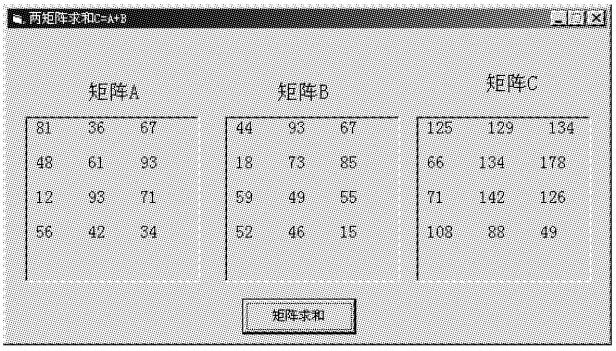

图 3-5　矩阵的加法计算结果图

【设计过程】

1. 界面设计

启动 Visual Basic 6.0，新建一个“标准 EXE”工程，然后在窗体 Form1 上绘制一个命令按钮(Command1)、3 个 Label 控件和 3 个 PictureBox 控件，设置各个对象的属性后，界面如图 3-6 所示。

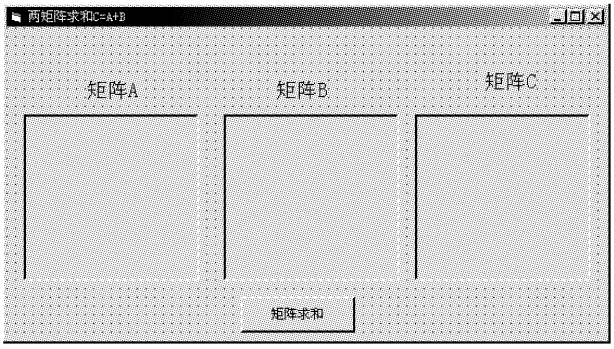

图 3-6　矩阵的加法界面设计图

2. 属性设置

在属性窗口里进行属性设置，如表 3-2 所示。

表 3-2　属性设置

对　象	属性名	属 性 值	对　象	属性名	属 性 值
Form1	Caption	"两矩阵求和 C＝A＋B"	Label3	Caption	"矩阵 C"
Label1	Caption	"矩阵 A"	Command1	Name	"cmdAdd"
Label2	Caption	"矩阵 B"	Command1	Caption	"矩阵求和"

3. 代码设计

双击命令按钮 cmdAdd 打开代码编辑器，在其 cmdAdd_Click 的事件中写入如下

代码：

```
Private Sub cmdAdd_Click()
    Dim a(4,3) As Integer,b(4,3) As Integer,c(4,3) As Integer
      Picture1.Cls
      Picture2.Cls
      Picture3.Cls
      For i=1 To 4
        For j=1 To 3
          a(i,j)=Int(Rnd * 91)+10
          b(i,j)=Int(Rnd * 91)+10
          c(i,j)=a(i,j)+b(i,j)
        Next j
      Next i

      For i=1 To 4
        For j=1 To 3
          Picture1.Print a(i,j);"";
          Picture2.Print b(i,j);"";
          Picture3.Print c(i,j);"";
        Next j
        Picture1.Print:Picture1.Print
        Picture2.Print:Picture2.Print
        Picture3.Print:Picture3.Print
      Next i
End Sub
```

【相关知识】

1. 二维数组的概念和定义

在计算机中二维数组其实只是一个逻辑上的概念，二维数组相当于一个方阵，每个元素需要两个下标来确定位置。在内存中，二维数组只按元素的排列顺序存放，形成一个序列。二维数组的应用很广，例如在案例3-2 中二维数组表示矩阵，两个下标分别与数据所处的行和列相对应，使矩阵中的每个元素都可在二维数组中找到相对应的存储位置。

二维数组的定义格式如下：

```
Dim 数组名(行标,列标)[As 类型]
```

例如案例 3-2 中语句：

```
Dim a(4,3)As Integer
```

定义了数组 a，具有 4 行 3 列来存储将要进行加法运算的矩阵，每一个数组元素的值都是整型数据。

2. 二维数组的赋值

对二维数组及多维数组的元素赋初值时，采用“按行优先”。赋值时可采用对元素全部赋值和单个赋值两种方式。全部赋值时一般运用双重循环语句，例如案例 3-2 中使用 for 语句为二维数组赋初值：

```
For i=1 To 4
    For j=1 To 3
      a(i,j)=Int(Rnd*91)+10
      b(i,j)=Int(Rnd*91)+10
      c(i,j)=a(i,j)+b(i,j)
    Next j
  Next i
```

案例 3-3　插入一个数到一个有序的数组中

【案例效果】

设计程序，用户通过键盘输入一个数，如图 3-7 所示，插入到一个递减的有序数组中，插入后使数组仍然递减有序。并在窗体上分别输出插入前后的数据。其效果如图 3-8 所示。

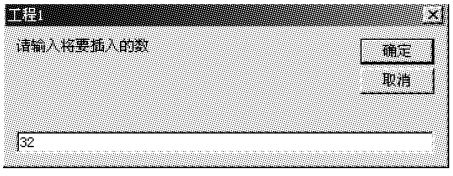

图 3-7　输入数据

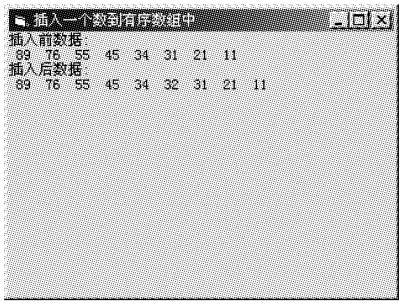

图 3-8　数据插入后的效果

【设计过程】

1. 界面设计

启动 Visual Basic 6.0，在“新建工程”对话框选择新建一个“标准 EXE”工程，新建一个 caption 为“插入一个数到有序数组中”的窗体。

2. 代码设计

双击窗体，打开代码编辑器，在窗体的 Form_Click 事件中写入如下代码：

```
Private Sub Form_Click()
    Dim a(),i%,n%,t%
```

```
    a=Array(89,76,55,45,34,31,21,11)
    n=UBound(a)
    Print"插入前数据："
    For i=LBound(a) To UBound(a)
      Print a(i);
    Next i
    Print
    ReDim Preserve a(n+1)
  t=Val(InputBox("请输入将要插入的数"))
  For i=UBound(a)-1 To LBound Step-1
      If t>=a(i) Then
         a(i+1)=a(i)
         If i=LBound Then a(i)=t
      Else
         a(i+1)=t
         Exit For
      End If
  Next i
  Print"插入后数据："
  For i=LBound(a) To UBound(a)
      Print a(i);
  Next i
End Sub
```

【相关知识】

1. 动态数组的概念

在有些情况下，用户可能不知道需要多大的数组，这时就需要定义一个能够改变大小的数组，这就是动态数组。动态数组可以在任何时间改变大小。在 Visual Basic 中，动态数组是最灵活、最方便的一种数组。利用动态数组还有助于有效管理内存，因为动态数组是使用时才开辟内存空间，在不使用这个数组时，还可以将内存空间释放给系统。这样就可以最大限度地节省内存，提高运行速度。

例如案例 3-3 中，插入数据后数组的长度会改变，我们定义了一个动态数组来存放数据。

2. 动态数组的定义和使用

创建动态数组需要两步。

(1) 和固定长度数组(静态数组)类似，用 Dim 语句(或 Private、Public、Static)定义，但是不要指定维数。如：

```
Dim MyArry()As Integer
```

(2) 以后的实际程序中，当要用到该数组时，再用 ReDim 语句分配实际的元素个数，

这时需要确定元素的个数。如前面声明的数组 MyArry，可以用下面语句将它定义为一个二维数组：ReDim MyArry(10,10)。

说明：

① ReDim 语句是一个可执行语句，只能出现在过程中，并且可以多次使用，改变数组的维数和大小。

② 定长数组定义时下标只能是常量，而动态数组 ReDim 语句中的下标可以是常量，也可以是有了确定值的变量。

③ 在过程中可以多次使用 ReDim 来改变数组的大小，也可改变数组的维数。例如：

```
ReDim x(10)
ReDim x(20)
x(20)=30
Print x(20)
ReDim x(20,5)
x(20,5)=10
Print x(20,5)
```

④ 每次使用 ReDim 语句都会使原来数组中值丢失，但可以在 ReDim 后加 Preserve 参数来保留数组中的数据。但此时只能改变最后一维的大小。

例如在案例 3-3 中，首先用 Dim a()定义一个动态数组 a，在使用过程中，语句 a=Array(89, 76, 55, 45, 34, 31, 21, 11)给数组赋初值，在插入数据之前，用 ReDim Preserve a(n+1)语句来改变数组的大小，ReDim 后加 Preserve 参数来保留数组中的数据。

3. 与数组操作相关的几个函数

(1) Array 函数。Array 函数可方便地对数组整体赋值，赋值后的数组大小由赋值的个数决定。但它只能给定义为变体的动态数组赋值。

例如，案例 3-3 中先用语句 Dim a()定义一个动态数组，用语句：

```
a=Array(89,76,55,45,34,31,21,11)
```

给数组赋初值。

注意：使用 Array 函数建立动态数组的下界由 Option Base 语句指定的下界决定，默认情况下为 0。

(2) LBound 与 UBound 函数。LBound 函数和 UBound 函数都是返回一个 Long 型数据，LBound 函数得到的值为指定数组维可用的最小下标，而 UBound 函数得到的是最大下标。使用形式如下：

```
UBound(<数组名>[,<N>])
LBound(<数组名>[,<N>])
```

其中参数含义如下。

＜数组名＞：数组变量的名称，遵循标准变量命名约定。

.<N>：可选的参数。指定返回哪一维的上界。1 表示第一维，2 表示第二维，如此等等。如果省略默认是 1。

例如，案例 3-3 中要输出动态数组 a 的各个值，用下面的语句进行了实现：

```
For i=LBound(a) To UBound(a)
    Print a(i);
Next i
```

(3) Option Base 语句。Option Base 语句在模块级别中使用，用来声明数组下标的默认下界。语法结构：

```
Option Base{0 1 1}
```

说明：默认状态下数组下界为 0，此时无须使用 Option Base 语句。如果使用该语句规定数组下界 1，则必须在模块的数组声明之前使用 Option Base 语句。Option Base 语句只影响位于包含该语句的模块中的数组下界。

案例 3-4　冒泡排序法

【案例效果】

设计程序，用户单击窗体时系统随机产生 50 个 10～90 之间的数，用冒泡排序法实现其从小到大排序，并将排序结果输出窗体上。其效果如图 3-9 所示。

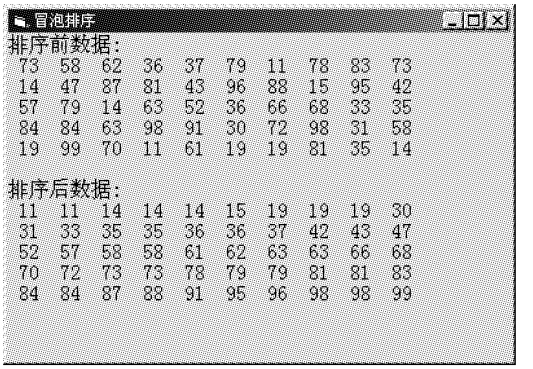

图 3-9　数据排序前后的效果图

【设计过程】

1. 界面设计

启动 Visual Basic 6.0，在“新建工程”对话框选择新建一个“标准 EXE”工程，新建一个 caption 为“冒泡排序”的窗体。