第5章 数 组

在编程时常常遇到需要对一系列同类型的数据进行处理的问题，如果使用前面所介绍的简单变量来处理这类问题会有很大困难。因为简单变量之间是相互独立、没有内在联系的，这就需要为不同的数据定义名称不同的变量，当数据量较大时，变量的定义和处理过程将会非常的复杂而繁琐。如果能够给一个数据集合中的数据起一个共同的名字，用一组连续的编号来区分集合中的不同数据，那么就可以通过数据集合的名字和编号来表示其中的每个数据。采用这样的方式表示数据将使得对大量数据的处理变得很方便。这样的一种具有共同名称、以编号来区别不同数据的数据集合就称为数组。

5.1 数组概述

数组是具有相同类型的有序变量的集合。这些变量按照一定的规则排列，占据着一段连续的存储单元。

5.1.1 数组的概念

数组名的命名规则与简单变量命名规则一样。数组名不是代表一个变量，而是代表有内在联系的一组变量，数组名中实际存放的内容是这组数据在内存中存放的首地址。数组内的每个成员称为数组元素，数组元素就是一个个简单变量，数组元素的类型也就是数组的类型。为了标识数组中的不同元素，每个数组元素都有各自的编号即下标，下标确定了数组元素在数组中的位置，可以用数组名和下标唯一地标识数组中的某个元素。

数组元素的一般表示形式如下：

```
数组名(下标1[,下标2,…])
```

其中，下标可以是常量、变量或算术表达式。当下标的值为非整数时，会自动进行四舍五入处理。

如果只需一个下标就可以确定一个数组元素在数组中的位置，则该数组称为一维数组。如果需要两个下标才能确定一个数组元素在数组中的位置，则该数组称为二维数组。以此类推，必须由 N 个下标才能确定一个数组元素在数组中的位置，则该数组称为 N 维数组。因此，确定数组元素在数组中位置的下标个数就是数组的维数。通常把二维以上

的数组称为多维数组，Visual Basic 规定数组的维数不得超过 60。

5.1.2 数组的定义

Visual Basic 规定在使用一个数组之前必须对数组进行定义，确定数组的名称和它的数据类型，指明数组的维数和每一维的上、下界的取值范围，这样系统就可以为数组分配一块连续的内存区域，存放数组的所有元素。数组的每个元素在这个连续的内存区域内都占据各自特定的单元。

在 Visual Basic 中有两种类型的数组：固定大小数组和动态数组。在定义数组时就确定了数组的大小，并且在程序运行过程中不能改变其大小的数组称为固定大小数组。在定义数组时不指明数组的大小，在程序运行时才根据需要确定其大小，即在程序运行中可以改变大小的数组称为动态数组。

1. 数组定义语句

数组定义语句的一般形式如下：

Public | Private | Static | Dim<数组名>([<维界定义>])[As<数据类型>]

说明：

(1) Public、Private、Static、Dim 是作用域关键字，数组作用域的含义与变量作用域的含义类似(作用域的概念将在第 6 章中重点介绍)，不同关键字的含义如表 5-1 所示。

表 5-1 数组作用域说明

关 键 字	适 用 范 围
Public	用于标准模块的声明段，定义全局数组
Private 和 Dim	用于模块的声明段，定义模块级数组
Dim	用于过程中，定义局部数组
Static	用于过程中，定义静态数组

(2) “维界定义”的格式如下：

[<下界 1>To] 上界 1 [[,<下界 2>To] 上界 2…]

格式中的下界 1、上界 1 表示数组第一维下标的下界和上界，下界 2 和上界 2 表示数组第二维下标的下界和上界，以此类推。“下界”和关键字“To”可以缺省，缺省情况下数组元素下标的取值是从 0 开始，等价于“0 To 上界”。如果代码中有 Option Basic 1 声明语句，则缺省的下界是从 1 开始，等价于“1 To 上界”。

例如，在窗体模块的“通用”部分有下列数组说明语句：

```
Dim a(3) As Integer,b(1 To 2,2) As Single
Private c(-6 To -2) As String * 6
```

第一条语句等价于：

```
Dim a(0 To 3) As Integer,b(1 To 2,0 To 2) As Single
```

它定义了一个模块级的一维整型数组 a 以及二维单精度型数组 b。一维数组 a 共有 4 个数组元素，分别是 a(0)、a(1)、a(2)、a(3)。二维数组 b 共有 6 个元素，分别是 b(1,0)、b(1,1)、b(1,2)、b(2,0)、b(2,1)、b(2,2)。

第二条语句定义了一个模块级的一维定长字符串型数组 c，数组的维下界是－6，维上界是－2，共有 5 个元素，分别是 c(－6)、c(－5)、c(－4)、c(－3)、c(－2)。每个元素都是一个 6 字符的定长字符串，占 6 个字节的内存空间，因此整个数组占 30 字节的内存空间。

2. 数组的上、下界

某维的下界和上界分别表示该维下标的最小值和最大值。维界的取值范围不得超过长整型(Long)数据表示的范围(－2 147 483 648～2 147 483 647)，且下界必须小于等于上界。在定义固定大小数组时，只能用常量或常量表达式进行维界说明。如果数组定义语句中的维界不是整数，将自动按照 CInt 函数的方法对其进行四舍五入取整。例如：

```
Dim x As Integer
Const N As Integer=10
Dim a(5.5) As Integer          '正确,该语句等同于 Dim a(1 To 6) As Integer
Dim b(1 To 3 * 5) As Integer   '正确,该语句等同于 Dim b(1 To 15) As Integer
Dim c(N) As Integer            '正确,该语句等同于 Dim c(10) As Integer
Dim d(0 To x) As Integer       '错误,不允许使用变量做数组的维界说明
```

注意：若用符号常量说明数组的维界，那么该符号常量在说明语句之前必须已定义过。

3. 数组的类型

数组定义语句中“As 数据类型”用来声明数组的类型，数据的类型即数组中元素的数据类型。数组的类型可以是 Integer、Long、Single、Double、Date、Boolean、String(变长字符串)、String * length(定长字符串)、Object、Currency、Variant 和自定义类型。若缺省 As 短语，则表示该数组是变体(Variant)类型。

4. 数组的大小

用数组说明语句定义数组，指定了各维的上、下界取值范围，也就确定了数组的大小。所谓数组的大小就是这个数组所包含的元素的个数。数组的大小有时也称为数组的长度，可用下面的公式计算得到：

数组的大小=第一维大小×第二维大小×…×第 N 维大小
各维的大小=维上界—维下界+1

例如，程序有下面的语句：

```
Option Base 1
Dim a(10) As Integer
Dim b(5,-3 To -1) As String
```

说明：Option Base 1 语句只能位于模块的通用部分，用以说明缺省下界的数组的元素下标默认是从 1 开始的。因此：

一维数组 a 的大小为 10－1＋1＝10 个数组元素。

二维数组 b 的大小为(5－1＋1)×[－1－(－3)＋1]＝5×3＝15 个数组元素。

5. 数组元素的初始值

数组说明语句不仅定义了数组的作用域，分配了存储空间，而且还对数组元素的值进行了初始化。数组元素的初始值与变量的初始值相同，与元素的类型有关。即数值型数组元素的初始值为零，变长字符型数组元素的初始值为空字符串，定长字符型数组元素的初始值为指定长度个数的空格，布尔型数组元素的初始值为“False”，变体型数组元素的初始值是“Empty”。

5.1.3 数组的结构

数组是具有相同数据类型的多个变量的集合，数组中的所有元素按一定顺序存储在连续的内存单元中。下面分别讨论一维/二维/三维数组的结构。

1. 一维数组的结构

一维数组的逻辑结构为一个线性队列。假设有如下语句：

```
Dim a(8) As Integer
```

则数组 a 的逻辑结构为：

```
a(0),a(1),a(2),…,a(6),a(7),a(8)
```

由于内存也是线性结构，因此，一维数组的逻辑结构与其在内存中存放的次序是一致的。

a(0) a(1) a(2) a(3) a(4) a(5) a(6) a(7) a(8)

2. 二维数组的结构

二维数组的逻辑结构为一张由行和列组成的二维表，与数学中的矩阵相同。二维数组的数组元素需要用两个下标来标识，即要指明数组元素的行号和列号。假设有如下语句：

```
Option Base 1
Dim t(3,4) As Integer
```

定义了一个二维数组 t，其中行列下标的下界都是从 1 开始，行下标的上界为 3，列下标的

上界为 4,因此这是一个 3 行 4 列的二维数组。该二维数组的逻辑结构示意如下:

	第 1 列	第 2 列	第 3 列	第 4 列
第 1 行	t(1,1)	t(1,2)	t(1,3)	t(1,4)
第 2 行	t(2,1)	t(2,2)	t(2,3)	t(2,4)
第 3 行	t(3,1)	t(3,2)	t(3,3)	t(3,4)

二维数组在内存中是"按列存放"在线性结构的内存中的。即先存放第一列的所有元素:t(1,1) t(2,1) t(3,1),接着存放第二列所有元素:t(1,2) t(2,2) t(3,2) ……直到存完最后一列的所有元素。二维数组 t 的元素在内存中的具体存放次序如下:

```
t(1,1) t(2,1) t(3,1) t(1,2) t(2,2) t(3,2) t(1,3) t(2,3) t(3,3) t(1,4) t(2,4) t(3,4)
```

3. 三维数组的结构

三维数组是由行、列和页组成的三维表。三维数组也可理解为几页的二维表,即每页由一张二维表组成。三维数组的元素是由行号、列号和页号共同来标识的。假设有如下语句:

```
Option Base 1
Dim p(2,3,2) As Integer
```

定义了一个三维数组 p,其中第一个数 2 标识了数组的行下标上界为 2,第二个数 3 标识了列下标上界为 3,第三个数 2 标识了页下标上界为 2。因此,这个三维数组有 2 页、2 行、3 列共 12 个元素。三维数组 p 的逻辑结构示意如下:

第 1 页	p(1,1,1)	p(1,2,1)	p(1,3,1)
	p(2,1,1)	p(2,2,1)	p(2,3,1)
第 2 页	p(1,1,2)	p(1,2,2)	p(1,3,2)
	p(2,1,2)	p(2,2,2)	p(2,3,2)

三维数组在内存中是按"逐页逐列"的规则存放的。即先对数组的第一页中的所有元素按列的顺序分配存储单元,然后再对第二页中的所有元素按列的顺序分配存储单元……直到数组的每一个元素都分配了存储单元。三维数组 p 的元素在内存中的具体存放次序如下:

```
p(1,1,1) p(2,1,1) p(1,2,1) p(2,2,1) p(1,3,1) p(2,3,1) p(1,1,2) p(2,1,2) p(1,2,2) p(2,2,2)
p(1,3,2) p(2,3,2)
```

5.2 数组的基本操作

由于数组元素的本质仍是变量,只不过是带有下标的变量而已,所以对数组元素的输入输出操作可以与变量相同。但与普通变量不同的是,数组元素的下标是有序排列的,可

以通过循环变量表示下标访问不同的数组元素。因此在需要对整个数组或数组中连续的元素进行处理时，利用循环结构是最有效的方法。

5.2.1 数组元素的赋值

1. 用赋值语句给数组元素赋值

单个数组元素的赋值可以通过赋值语句来实现。例如：

```
Dim a(3) As Integer,b(1 To 2,2) As Single
a(0)=10
b(1,0)=1
b(2,2)=5
```

2. 通过循环逐一给数组元素赋值

若在一个 For 循环中用循环控制变量作为数组元素的下标，就可依次访问一维数组的每一个元素。同样使用双重的 For 循环，用外层、内层循环的循环控制变量分别作为第一维、第二维的下标就可依次访问二维数组的所有元素。以此类推，数组有 N 维就可以采用 N 重循环给数组的所有元素一一赋值。

例如，下面程序的功能是：在窗体输出一个两位随机数构成的一维数组；在图片框上输出一个三位随机数构成的二维数组。请注意代码中循环控制变量作为数组元素下标的使用方法。程序的运行结果如图 5-1 所示。

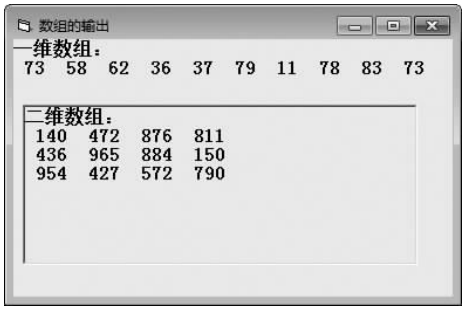


图 5-1 数组赋值输出

```
Option Explicit
Option Base 1
Private Sub Form_Click()
    Dim a(10) As Integer,b(3,4) As Integer
    Dim i As Integer,j As Integer
    Print "一维数组："
    For i=1 To 10                    '使用循环给一维数组赋值并输出
        a(i)=Int(90 * Rnd)+10
        Print a(i);                  '分号用于在一行上连续输出数组元素
    Next i
    Print                            '在窗体上换行
    Picture1.Print "二维数组："
    For i=1 To 3                     '利用二重循环给二维数组赋值并输出
        For j=1 To 4
            b(i,j)=Int(900 * Rnd)+100
            Picture1.Print b(i,j);
```

```
        Next j
        Picture1.Print          '在图片框中换行
    Next i
End Sub
```

3. 用 InputBox 函数给数组元素赋值

用 InputBox 函数可以实现通过弹出的输入框给数组元素赋值。例如，下面程序的功能是：用 InputBox 函数给数组元素赋值，并将数组元素以每行 6 个的形式输出到文本框中。InputBox 上有动态的提示信息，即用户随时知道在录入第几个元素的值。程序运行界面如图 5-2 所示。

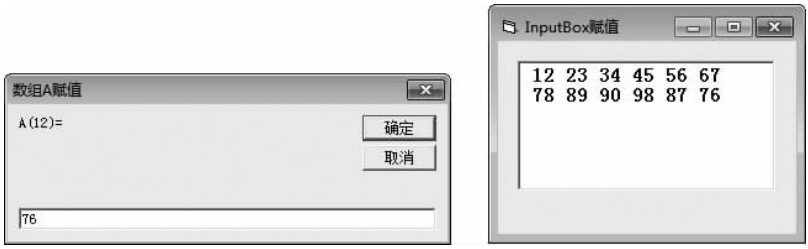


图 5-2　InputBox 函数给数组元素赋值

```
Option Explicit
Option Base 1
Private Sub Form_Click()
    Dim a(12) As Integer,i As Integer,st As String
    For i=1 To 12
        a(i)=Val(InputBox("A(" & i & ")=","数组 A 赋值"))
        st=st & Str(a(i))
        If i Mod 6=0 Then          ' 每行显示 6 个数据
            st=st & vbCrLf         '这里也可通过 Chr(13) & Chr(10)实现换行
        End If
    Next i
    Text1=st
End Sub
```

注意：本程序界面中文本框的 MultiLine 属性必须设置为 True，否则无法实现多行显示。

小提示：利用 InputBox 函数从键盘输入多个数值时，可以利用数字键区，输完数字后按 Enter 键进行确认，这样可以提高录入数据的速度。

由于在执行 InputBox 函数时程序会暂停运行等待输入，并且每次只能输入一个值，占用运行时间长，所以 InputBox 函数只适合输入少量数据。如果数组比较大，需要输入的数据较多，用 InputBox 函数给数组赋值就显得不高效，这时可以运用文件操作从文件中批量地读取和保存数据(参见教材第 8 章中有关文件的知识)。

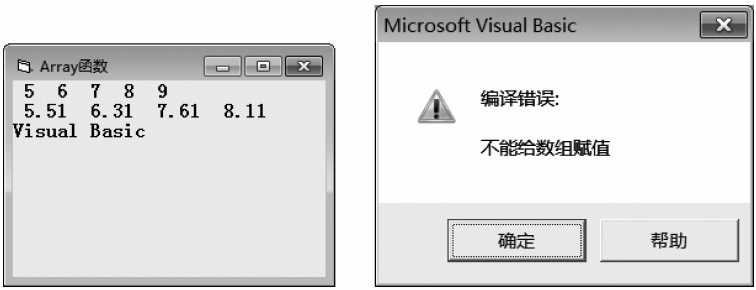
4. 用 Array 函数给数组赋值

利用 Array 函数可以把一个数据集赋值给一个 Variant 变量，再将该 Variant 变量创建成一个一维数组。Array 函数的一般使用形式如下：

```
<变体变量名>=Array([数据列表])
```

注意：Array 函数只能给 Variant 类型的变量赋值。“数据列表”是用逗号分隔的赋给数组各元素的值。

函数创建的数组的长度与列表中的数据的个数相同。若缺省数据列表，则创建一个长度为 0 的数组。若程序中缺省 Option Base 1 语句或使用了 Option Base 0 语句，则 Array 函数创建的数组的下界从 0 开始；若窗体的通用部分有 Option Base 1 语句，数组的下界从 1 开始。例如执行如下程序，运行结果见图 5-3(a)。



(a) (b)

图 5-3 Array 函数的使用

```
Option Explicit
Option Base 1
Private Sub Form_Click()
    Dim a As Variant
    Dim b(4) As Variant
    a=Array(5,6,7,8,9)                    'a 包含整型数组
    Print a(1); a(2); a(3); a(4); a(5)
    a=Array(5.51,6.31,7.61,8.11)          'a 包含单精度型数组
    Print a(1); a(2); a(3); a(4)
    a="Visual Basic"                      'a 成为字符型变量
    Print a
    b=Array(1,2,3,4,5)                    '该语句有错误
End Sub
```

运行该程序，执行语句“a＝Array(5,4,3,2,1)”，Array 函数就创建了一维数组 a，数组元素的类型是 Integer。该数组的下标从 1 开始，共有 a(1)、a(2)、a(3)、a(4)、a(5)等 5 个元素，它们的值分别是 5、4、3、2、1。这里的 a 是一个包含数组的 Variant 变量，与类型是 Variant 的数组是完全不相同的，可再次用赋值语句“a＝Array(1.51,2.31,3.61,4.11)”给 a 赋值，此刻 Array 函数创建的数组元素个数是 4，数组元素的类型改为 Single。

也可以用普通的赋值语句给已包含数组的 Variant 变量 a 赋一个值，例如，a＝"NO Array"。执行该语句后，a 不再包含数组，又成为一个普通的字符型变量。当执行语句“b＝Array(1,2,3,4,5)”时，会产生一个“给数组赋值”的错误，如图 5-3(b)所示，其原因是 b 目前已经是被定义为 Variant 类型的数组，而不是一个普通的 Variant 类型的变量。

注意：不可以用 Array 函数给非 Variant 类型的变量赋值。

5. 用文本框数据给数组赋值

对大批量的数据输入，采用文本框输入效率更高。输入时可以采用 Instr 函数获取分隔符的位置从而给数组元素赋值，也可以采用 Split 函数方便地给数组赋值。Split 函数返回一个下标从零开始的一维数组，赋值号左边必须是一个变体型变量。以下程序实现了用文本框给数组赋值，其运行结果如图 5-4 所示。

```
Option Explicit
Private Sub Command1_Click()
    Dim a As Variant,i As Integer
    a=Split(Text1,",")                          '逗号为数据分隔符
    For i=LBound(a) To UBound(a)
        Picture1.Print "A("; i; ")="; a(i)
    Next i
End Sub
```

如果在文本框 Text1 中输入“27,584,987,21.68”，则运行程序后 a 会变成一个含有 5 个元素的数组，分别是 a(0)＝27，a(1)＝584，a(2)＝987，a(3)＝21.68。

图 5-4　Split 函数获取数组元素

图 5-5　Instr 函数获取数组元素

以上代码也可通过 Instr 函数找到分隔符的位置，然后将文本框内容分隔后输入到数组 a，运行结果如图 5-5 所示。

```
Option Explicit
Private Sub Command1_Click()
    Dim a(4) As Integer,i As Integer,n As Integer,s As String
    s=Text1
    For i=0 To 2
        n=InStr(s,",")                          ' 逗号为数据分隔符
```

```
        a(i)=Left(s,n-1)
        Picture1.Print "a("; i; ")="; a(i)
        s=Right(s,Len(s)-n)
    Next i
    a(i)=s                                          '给最后一个数组元素赋值
    Picture1.Print "a("; i; ")="; a(i)
End Sub
```

当数据较多时,可在设计状态下将文本框的 Text 属性设置为输入数据。如果数据是以别的符号分隔,可以将上述程序中的分隔符适当修改即可。

5.2.2 数组元素的输出

在程序中可以像使用普通变量一样引用数组元素,也就是说,数组元素可以出现在表达式中的任何位置。在引用数组元素时,数组元素的下标表达式的值一定要在定义数组时规定的维界范围之内,否则就会产生"下标越界"的错误。

数组元素的输出与普通变量的输出完全相同,即可以使用 Print 方法将数组元素显示在窗体上或图片框中,也可将数组元素显示到文本框、列表框中。程序调试时还可以用 Debug.Print 将数组元素显示到"立即"窗口中。

输出数组元素的方法与输入类似,整体输出时通常利用循环结构实现。

【例 5-1】 产生 10 个两位随机整数打印在窗体上,求出其中的最大值和最小值并输出它们所在的位置。

算法分析:用两个变量 Max、Min 分别记录数组的最大值和最小值,依次用数组元素的值与记录最大值和最小值的变量 Max、Min 比较,当数组元素值大于 Max 时就将元素值放进 Max,同时用变量 m 记录其所在的位置;当数组元素值小于 Min 时就将元素值放进 Min,同时用变量 n 记录其所在的位置。在循环过程中始终保持 Max、Min 中的值是已参与比较的数组元素中的最大值和最小值。用一维数组配合一重 For 循环就可以实现求数组元素的最大和最小值。运行结果见图 5-6。

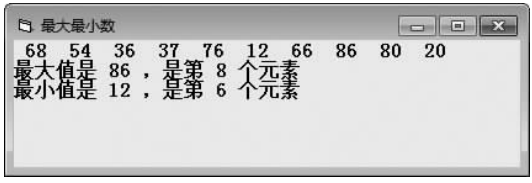


图 5-6 求最大最小值运行界面

```
Option Explicit
Option Base 1
Private Sub Form_click()
    Dim Compare(10) As Integer,i As Integer
    Dim Max As Integer,Min As Integer
    Dim n As Integer,m As Integer
```