

当需要处理的数据具有同一性质并且数据量大时，例如要记录一个班的 30 名同学的成绩；要做一个上千人的职工信息表；输入 100 个数，然后排序，等等，如果用变量存储数据会很不方便，这时用数组实现比较方便。数组是在基本数据类型的基础上扩展的数据类型，对于构思比较复杂的解题思路是非常有帮助的。数组是编程语言数据类型中重要的数据结构。数组与循环结构结合使用可以有效地处理大批量具有相同类型的数据，解决用简单变量无法（或困难）实现的问题。

5.1 数 组 概 述

1. 数组的定义

数组是一组具有相同数据类型的变量的集合。每个数组都有一个名字，数组名代表逻辑上相关的一批数据，数组中元素的个数称为数组的长度（大小）。数组在内存中被分配连续的存储单元，每个存储单元代表数组中的数据，数组中的每一个数据称为数组元素，用数组名和该数据在数组中的序号来标识。序号又称为下标，数组元素又称为下标变量。数组元素由数组名、一对括号和下标来表示，数组元素可以像普通变量一样使用。利用数组可以简化程序，提高编程效率。

2. 数组的特点

- (1) 数组能保存多个值。
- (2) 数组是相同数据类型的元素的集合。
- (3) 数组可以与循环语句配合使用，实现复杂的运算。
- (4) 数组元素用整个数组的名字和它自己在数组中的顺序位置来表示。例如， $a[0]$ 表示名字为 a 的数组中的第一个元素， $a[1]$ 代表数组 a 的第二个元素，依此类推。
- (5) 数组常用来表示与一维、二维、三维空间分布相关的数据。一维数组相当于数学中的数列，二维数组相当于数学中的矩阵。
- (6) 动态数组可以根据需要开辟内存空间，优化程序，提高效率。
- (7) 数组还可以用作过程或函数的参数或返回值，可传递或返回大量的值。
- (8) 数组中的各元素是有先后顺序的。

3. 数组的分类

在 Visual Basic 中，数组一般可以从以下几个角度进行分类。

- (1) 按数组所占存储空间的不同分类。

① 静态数组：声明时确定了大小的数组，所占的内存空间是固定不变的。

② 动态数组：声明时没有给定数组大小，所占的内存空间是可变的。

(2) 按数组的维数分类。

① 一维数组：数组中的元素只有一个下标。

② 二维数组：数组中的元素有两个下标。

③ 多维数组：数组中的元素具有多个下标。

(3) 按数组的元素的数据类型分类。

① 数值型数组：数组元素是 Integer、Long、Single 等类型的变量。

② 字符串数组：数组元素是 String 类型的变量。

③ 日期型数组：数组元素是 Date 类型的变量。

④ 变体数组：数组元素是 Variant 类型的变量。

⑤ 自定义数组：数组元素是自定义类型的变量。

(4) 按数组的功能分类。

① 变量数组：数组元素都是同类型的变量。

② 控件数组：数组元素都是同类型的对象。

在 Visual Basic 中数组没有隐式声明，所有使用的数组必须先声明，后引用。数组的声明包括声明数组名、数组的维数及数据类型。

5.2 静态数组

静态数组是指程序在运行时数组元素的个数不变，所占有的分配空间也保持不变的数组。静态数组在声明时就已确定大小。静态数组分为一维数组、二维数组和多维数组。

5.2.1 一维数组

1. 一维数组的声明

只有一个下标的数组称为一维数组。其声明格式如下：

Public|Private|Dim|Static 数组名(<下标>) [As <类型>]

说明：

(1) **Public**：用于建立公用数组，在模块的声明段用 **Public** 声明数组，此类数组可在多个模块中被调用。

(2) **Private** 与 **Dim**：都可用于建立模块级数组或局部数组。

(3) **Static**：只能用于建立局部数组，在过程内部使用。

(4) 数组名：与变量的命名规则相同，遵循标识符命名规则。数组的作用域和变量的作用域相同。

(5) 下标：用来指定数组元素的个数（数组的长度），它是整型常量，不能是变量。

下标形式为 “[下界 To]上界”，当省略下界时，其默认值为 0。

数组大小为 “上界-下界+1”。

(6) **As<类型>**：指定数组（元素）类型，可以是 VB 语言支持的数据类型，如果省略则定义为变体类型，数组可以存储任意类型的数据。

例如：

```
Dim a(1 To 3) As Integer, b(4) As Integer
Dim c(-2 To 2)
Dim s(5) As String*8
```

第一条语句定义了两个整型数组 a 和 b , a 数组的长度为 3 (含 3 个元素), 元素为 $a(1)$ 、 $a(2)$ 、 $a(3)$, b 数组的长度为 5 (含 5 个元素), 元素为 $b(0)$ 、 $b(1)$ 、 $b(2)$ 、 $b(3)$ 、 $b(4)$, 两个数组的元素均为整型。

这里以 b 数组为例, 系统在内存中为 b 数组分配 5 个连续的存储单元, 每个存储单元占两个字节, 如图 5.1 所示。

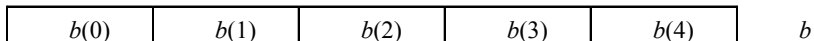


图 5.1 数组 $b(4)$ 的存储单元

第二条语句定义了类型为 **Variant** 的 c 数组, 其长度为 5, 元素为 $c(-2)$ 、 $c(-1)$ 、 $c(0)$ 、 $c(1)$ 、 $c(2)$, 数组的元素均为 **Variant** 型。

第三条语句定义了字符串类型的 s 数组, 有 6 个元素, 每个元素最多存放 8 个字符。

提示：

(1) 数组下界默认为 0, 为了便于使用, 在 VB 中的窗体层或标准模块层用 “Option Base n ” 语句指定在定义数组时下标下界省略时的默认值 (不能在过程中使用)。

例如：

```
Option Base 1
Dim m(3) As Integer
```

设定默认下界为 1, 数组 m 中的元素分别为 $m(1)$ 、 $m(2)$ 、 $m(3)$ 。

注意：该语句只能对本模块中声明时默认下界的数组起作用, 对其他模块的数组不起作用。

(2) 在定义数组时可用类型符来指定数组的类型。例如：

```
Dim a(3) As Integer
Dim a%(3)
```

上述两条语句定义的整型数组是等价的。类似地还有：

```
Dim B!(3 To 5)、C#(12)、D$(5)
```

2. 数组元素的引用

一维数组被定义之后便具有了内存空间, 可以通过以下方式引用数组指定下标的元素。

数组名 (下标值)

引用数组元素时的“下标”可以是整型（或长整型）常量、变量或表达式。下标值不能小于数组下标的下界，不能大于下标的上界。

3. 数组元素的赋值

对于静态数组，数组声明后由于其个数是固定的，因此所有元素都已经在内存中分配好了空间，此时的数组元素与简单变量一样存在默认的初值，若需要给数组的元素重新赋值，可以采用直接赋值，当数组元素很多时也可以采用循环方式，使赋值简单、快捷。

1) 直接赋值

直接赋值即直接将值赋给数组中的每个元素。例如，下列程序对数组元素赋值如下：

```
Dim stu(3) As Integer
stu(0)=1
stu(1)=2
stu(2)=3
stu(3)=4
```

这就实现了对数组中的每个元素进行赋值。

2) 利用循环结构赋值

当需要赋值的数组元素较多，且所赋值有一定的规律时，使用循环结构赋值是非常有效的一种方法。例如，给数组 `stu(1 To 10)` 中的每个元素都赋初值 0，程序如下：

```
Dim stu(1 To 10) As Integer
Dim i As Integer
For i=1 To 10
    stu(i)=0
Next i
```

3) 数组元素间互相赋值

例如将 *A* 数组的元素值赋给 *B* 数组，对应的赋值程序为：

```
Dim A(1 To 100) As Integer, i As Integer
Dim B(1 To 100) As Integer
For i = 1 To 100
    A(i) = i
    Print A(i);
Next i
Print
For i = 1 To 100
    B(i) = A(i)
    Print B(i);
Next i
```

在 Visual Basic 6.0 中可以直接将一个数组的值赋给另一个数组。例如：

```
Dim stu(3) As Integer
Dim tea() As Integer
```

```
tea=stu
```

使用这种方法赋值需要注意以下事项:

(1) 赋值号两边的数组类型必须一致。

(2) 如果赋值号左边是一个动态数组,则赋值时系统自动将动态数组 ReDim 成与右边相同大小的数组。

(3) 如果赋值号左边是一个大小固定的数组,则数组赋值出错。

4) 利用 Array 函数赋值

例如,以下程序将两组数据转化为数组中的数组元素。

```
Dim a As Variant, b As Variant, i%  
a=Array(1,2,3,4,5)  
b=Array("abc","def","67")
```

注意: 本函数只能给变体类型的变量或变体类型的动态数组赋值。

4. 数组元素的输出

数组元素的输出方法与简单变量的输出方法一样,在对多个数组元素进行输出时也可以采用循环方式,使得数组元素的输出更为简单、快捷。例如从键盘输入 10 个整数,存放在一维数组中并按逆序输出,则对应的输出程序如下:

```
Private Sub Command1_Click()  
    Dim a(10) As Integer, i%  
    Print "输入的数据为:"  
    For i = 1 To 10  
        a(i) = InputBox("请输入一个整型数")  
        Print a(i);  
    Next i  
    Print  
    Print "逆序输出为"  
    For i = 10 To 1 Step - 1  
        Print a(i);  
    Next i  
End Sub
```

【例 5.1】 由键盘输入 10 名学生的成绩,求出最高分和最低分。

解题思路: 将学生的成绩保存在一维数组 $a(1 \text{ To } 10)$ 中,设变量 $\max$ 中保存最高分,其初值为数组中的第一个元素 $\max=a(1)$,然后依次将 $\max$ 与 $a(2)$ 到 $a(10)$ 进行比较,如果数组中的某个数 $a(i)$ 大于 $\max$,则用该数替换 $\max$,即 $\max=a(i)$,所有数据比较完后, $\max$ 中存入的数即为数组中的最大值。求最小值的方法与求最大值的方法类似。

创建应用程序的用户界面和设置对象属性。在窗体上建立两个标签、两个文本框和一个命令按钮。标签 Label1 的 Caption 属性为“最高分:”,Label2 的 Caption 属性为“最低分:”。两个文本框的名称分别为 Text1 和 Text2,Text 属性均为空。命令按钮名称为 Command1,其 Caption 属性为“判断”。

单击“判断”按钮后通过键盘输入 10 个成绩,开始查找最高分和最低分,找到后分

别显示在文本框 Text1 和 Text2 上，并将输入的 10 名同学的成绩显示在窗体上。程序运行结果如图 5.2 所示，其中图 5.2 (a) 所示为单击“判断”按钮之前，图 5.2 (b) 所示为输入第 1 名同学的成绩，图 5.2 (c) 所示为输入第 10 名同学的成绩，图 5.2 (d) 所示为最终运行结果。

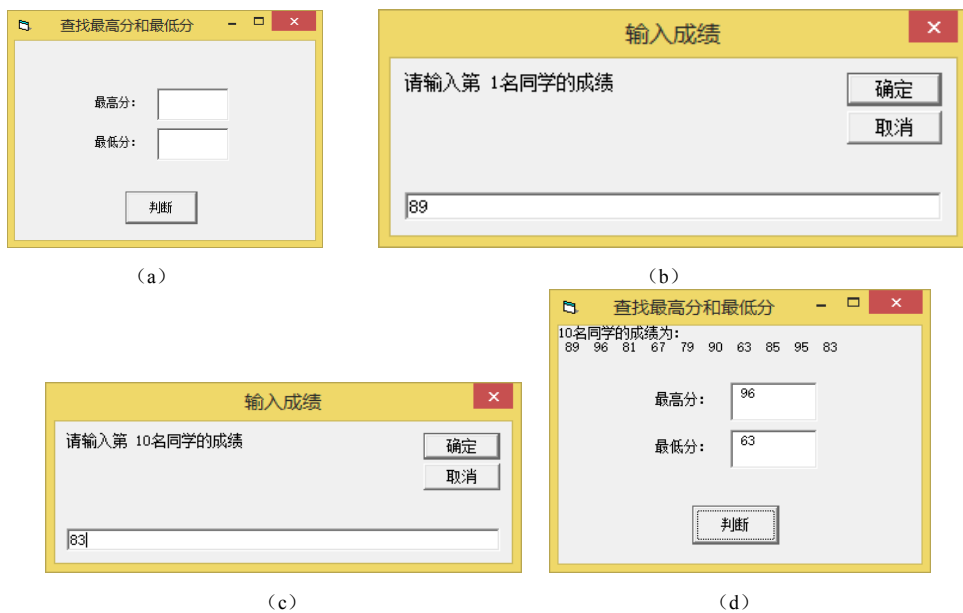


图 5.2 例 5.1 的运行结果

程序代码如下：

```
Private Sub Command1_Click()
    Dim a(1 To 10) As Integer, i As Integer
    Dim max As Integer, min As Integer
    For i = 1 To 10
        a(i) = Val(InputBox("请输入第" & Str(i) & "名同学的成绩", "输入成绩"))
    Next i
    Print "10名同学的成绩为: "
    For i = 1 To 10
        Print a(i);
    Next i
    max = a(1)
    min = a(1)
    For i = 2 To 10
        If max < a(i) Then max = a(i)
        If min > a(i) Then min = a(i)
    Next i
    Text1.Text = Str(max)
    Text2.Text = Str(min)
End Sub
```

【例 5.2】 计算 Fibonacci 数列的前 10 项之和。

解题思路： Fibonacci 数列从第 3 项开始，每一项等于前面两项之和，即 $f(i)=f(i-1)+$

$f(i-2)$ ，通过循环累加实现计算。程序运行结果如图 5.3 所示。

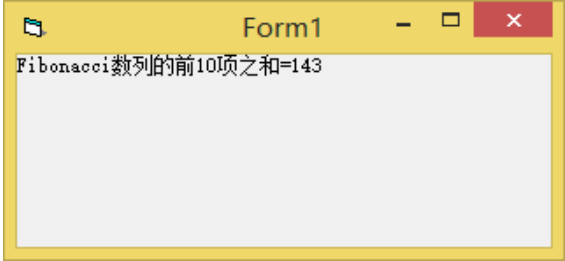


图 5.3 例 5.2 的运行结果

程序代码如下：

```
Private Sub Form_Click()  
    Dim i%, s%, f%(9)  
    f(0) = 1: f(1) = 1           ' 初始化数列的前两项  
    s = 2                       ' 前两项之和为s=2  
    For i = 2 To 9  
        f(i) = f(i - 1) + f(i - 2)   ' 每一项等于前面两项之和  
        s = s + f(i)  
    Next i  
    Print "Fibonacci数列的前10项之和=" & s  
End Sub
```

【例 5.3】 将十进制整数转换为二进制数。

解题思路：将十进制整数转换成二进制数的方法是“除 2 取余倒排列法”，即把要转换的十进制数反复除以 2 取其商，直到商为 0；再将余数倒排列即为对应的二进制数。在程序中使用了数组 b 保存每次除以 2 所得的余数。

创建应用程序的用户界面和设置对象属性。在窗体上建立两个标签、两个文本框和一个命令按钮。标签 Label1 的 Caption 属性为“十进制：”，Label2 的 Caption 属性为“二进制：”。两个文本框的名称分别为 Text1 和 Text2，Text 属性均为空。命令按钮的名称为 Command1，其 Caption 属性为“转换”。程序运行后，当用户在“十进制：”文本框中输入数据后，单击“转换”按钮，则在“二进制：”文本框中输出结果。程序运行结果如图 5.4 所示。

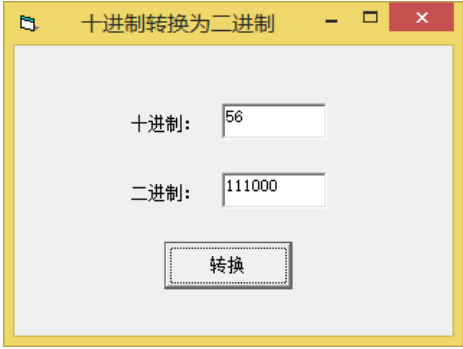


图 5.4 例 5.3 的运行结果

程序代码如下：

```
Option Base 0
Private Sub Command1_Click()
    Dim i As Integer
    Dim d As Integer      ' 表示十进制数
    Dim b(15) As Byte     ' 用于存储二进制数符
    Dim s As String
    d = Text1.Text        ' 得到十进制数
    Do Until d = 0
        b(i) = d Mod 2    ' 除2取余
        d = d \ 2
        i = i + 1
    Loop
    Do While i > 0
        i = i - 1
        s = s & b(i)      ' 反序排列
    Loop
    Text2.Text = s        ' 显示二进制数
End Sub
```

5.2.2 二维数组

1. 二维数组的声明

有两个下标的数组称为二维数组。二维数组的定义和一维数组的基本一样，只不过多了一个下标，其格式如下：

Public | Private | Dim | Static 数组名(<下标1>,<下标2>) [**As** <类型>]

说明：

- (1) 数组名、下标和类型与一维数组的规定相同。
- (2) 数组的大小为各维大小的乘积，行、列下标默认从 0 开始。

例如：

```
Dim a(2, 3) As Single          ' 元素个数为12 (3×4)
Dim b(1 To 3, 2 To 6) As Integer ' 元素个数为15 (3×5)
```

(3) 习惯上经常把数组的第一维称为行，第二维称为列，把一个二维数组和一个矩阵对应起来。二维数组在内存中的存放顺序是“先行后列”。

例如，数组 a 中的各元素在内存中的存放顺序如下：

$a(0,0) \rightarrow a(0,1) \rightarrow a(0,2) \rightarrow a(0,3) \rightarrow a(1,0) \rightarrow a(1,1) \rightarrow a(1,2) \rightarrow a(1,3) \rightarrow a(2,0) \rightarrow a(2,1) \rightarrow a(2,2) \rightarrow a(2,3)$

其逻辑结构如图 5.5 所示。

	第 0 列	第 1 列	第 2 列	第 3 列
第 0 行	$a(0,0)$	$a(0,1)$	$a(0,2)$	$a(0,3)$
第 1 行	$a(1,0)$	$a(1,1)$	$a(1,2)$	$a(1,3)$
第 2 行	$a(2,0)$	$a(2,1)$	$a(2,2)$	$a(2,3)$

图 5.5 数组 a 的逻辑结构

2. 数组元素的引用

二维数组元素的引用和一维数组的基本相同，二维数组中的元素可以用以下格式引用。

数组名(下标1, 下标2)

例如：

```
a(1,2)=20
a(i+2,j)=a(1,2)*3
```

3. 数组元素的赋值

在程序中经常采用双重循环的方法对二维数组赋值，例如从键盘输入 15 个整数并存到一个 3×5 的二维数组中，则实现的程序为：

```
Private Sub Command1_Click()
    Dim a(1 To 3, 1 To 5) As Integer, i As Integer, j As Integer
    For i = 1 To 3
        For j = 1 To 5
            a(i, j) = InputBox("请输入一个整数：")
        Next j
    Next i
End Sub
```

4. 数组元素的输出

在设计应用程序时，用双重循环将二维数组中的元素进行输出，可以使二维数组按行、列排序。例如一个 2×5 的二维数组中已经存有数据，现在将其输出，实现程序为：

```
Private Sub Command1_Click()
    Dim a(1 to 2,1 to 5) As Integer, i As Integer, j As Integer
    For i = 1 To 2
        For j = 1 To 5
            Print a(i,j);
        next j
        print
    Next i
End Sub
```

【例 5.4】 实现矩阵相加，即 $C=A+B$ （也就是二维数组对应元素相加）。程序运行结果如图 5.6 所示。

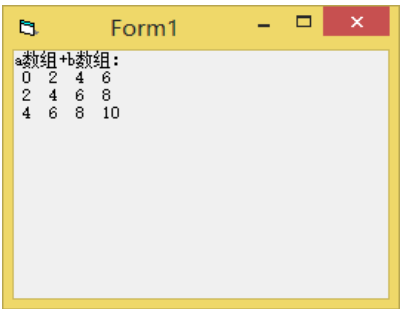


图 5.6 例 5.4 的运行结果

程序代码如下：

```
Private Sub Form_Click()  
    Dim a%(2, 3), b%(2, 3), c%(2, 3), i%, j%  
    For i = 0 To 2  
        For j = 0 To 3  
            a(i, j) = i + j  
            b(i, j) = i + j  
            c(i, j) = a(i, j) + b(i, j) ' 每个元素=下标之和  
            ' 对应元素相加  
        Next j  
    Next i  
    Print "a数组+b数组: "  
    For i = 0 To 2  
        For j = 0 To 3  
            Print c(i, j);  
        Next j  
        Print  
    Next i  
End Sub
```

【例 5.5】 打印 4 名同学的英语、数学、计算机 3 门课的考试成绩，并计算出每个同学的平均成绩。程序运行后单击窗体，依次输入学生姓名及其各门课程的成绩，运行结果如图 5.7 所示。



图 5.7 例 5.5 的运行结果

解题思路：把 4 名同学的姓名及各科的考试分数分别存入一个一维字符串数组 *xm* 和一个二维数值数组 *a* 中，然后对数组（主要是二维数组）进行处理。

程序代码如下：

```
Private Sub Form_Click()  
    Dim a(4, 3) As Single, xm(4) As String * 10, i%, j%, aver!  
    Print Tab(25); "成绩表"  
    Print  
    Print "姓名"; Tab(15); "英语"; Tab(25); "数学";  
    Print Tab(35); "计算机"; Tab(45); "平均分"  
    Print  
    For i = 1 To 4  
        aver = 0  
        xm(i) = InputBox("输入姓名")  
        Print xm(i);  
        For j = 1 To 3  
            a(i, j) = InputBox("输入" & xm(i) & "的一个成绩 ")  
            aver = aver + a(i, j)  
        Next j  
        aver = aver / 3  
        Print Tab(15); a(i, 1); Tab(25); a(i, 2);  
        Print Tab(35); a(i, 3); Tab(45); aver  
        Print  
    Next i  
End Sub
```

【例 5.6】 求二维数组中的最小值及其所在的行号、列号。

解题思路：假设一个二维数组 *a* 有 *N* 行 *N* 列（下界设为 1），用变量 *min* 来存放最小值，可以先把 *a*(1,1) 赋值给 *min* 变量，然后通过二重循环（循环控制变量 *i* 控制行标变化，*j* 控制列标变化），依次将 *min* 与 *a* 数组的每个变量 *a*(*i,j*) 进行比较，只要 *a*(*i,j*) < *min* 为 True 就把 *a*(*i,j*) 赋值给 *min*，并同时记下 *i* 和 *j* 的当前值，可用两个变量 *row* 和 *line*，令 *row*=*i* 且 *line*=*j* 实现。程序运行结果如图 5.8 所示。

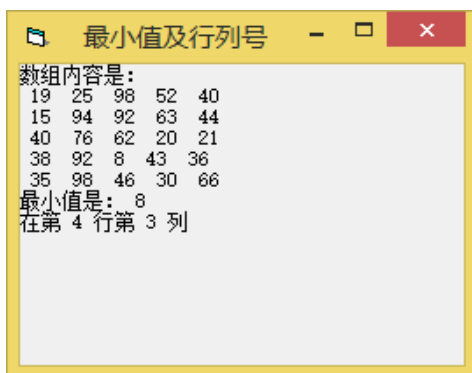


图 5.8 例 5.6 的运行结果

程序代码如下：

```
Option Base 1
Private Sub Form_Click()
    Randomize
    Const N = 5 ' 常数N确定数组的大小
    Dim a(N, N) As Integer
    Dim i, j, row, line, min As Integer
    For i = 1 To N ' For循环中i从1变化到N
        For j = 1 To N ' For循环中j从1变化到N
            a(i, j) = Int(100 * Rnd) + 1 ' 产生N*N个1~100的随机数
        Next j
    Next i
    ' 输出二维数组
    Print "数组内容是: "
    For i = 1 To N
        For j = 1 To N
            Print a(i, j);
        Next j
        Print ' 输完一行元素后输出一个换行
    Next i
    ' 设定初始值
    min = a(1, 1): row = 1: line = 1
    ' 寻找最小数
    For i = 1 To N
        For j = 1 To N
            If a(i, j) < min Then
                min = a(i, j)
                row = i
                line = j
            End If
        Next j
    Next i
    ' 输出结果
    Print "最小值是: "; min
    Print "在第"; row; "行第"; line; "列"
End Sub
```

5.2.3 多维数组

通常把数组的下标个数多于 3 个以上的数组称为多维数组。其一般定义格式如下：

Public | Private | Dim | Static 数组名 (<下标1>, <下标2>, ...) [**As** <类型>]

说明：

(1) 数组名、下标和类型与二维数组的规定相同。

(2) 数组的大小为各维大小的乘积，每一维的大小为“上界-下界+1”。

例如，下面定义的是两个多维数组。

```
Dim a(3,3,4) As Integer           ' 三维数组
Dim b(1 To 10,-4 To 5, 10,20) As Single ' 四维数组
```

上面的三维数组 *a* 有 80 个元素，四维数组 *b* 有 23 100 个元素。

多维数组是在一维和二维数组概念上的扩展。多维数组元素的引用、输入和输出与一维和二维数组类似。

用户可以将二维数组比作二维表格，那么三维数组可看作是由多张二维表组成的三维表格。维数大于 3 以上的多维数组在 VB 中很少使用。

5.3 动态数组

通过前两节的学习，读者知道定义数组是用数值常数或符号常量定义数组的维数及下标的上、下界。VB 编译程序在编译时为数组分配了相应的存储空间，并且在应用程序运行期间数组一直占据这块内存区域。但有时在程序运行前无法确定一个数组的大小及维数，数组应定义多大要在程序运行时才能决定。如果定义的数组过大，显然会造成内存空间的浪费，在这种情况下使用动态数组来处理就比较方便。动态数组是指在程序运行时可以增加或减少其元素个数的数组。

使用动态数组的优点是可以根据需要有效地利用存储空间（存储空间根据需求变大或变小）。

动态数组与静态数组的区别如下：

(1) 静态数组是系统在编译时根据声明语句定义的数组大小确定数组的长度预先分配存储空间，在程序执行过程中存储空间的大小不能改变，当程序执行结束时系统回收分配的空间。

(2) 动态数组声明语句未给出数组的大小（省略下标），在程序执行过程中根据需要可以多次执行 ReDim 语句，确定数组大小，动态分配存储空间。

5.3.1 动态数组的声明

声明动态数组的方法是首先使用 Public、Private、Dim 或 Static 语句声明一个没有下标的数组，即括号内为空的数组，然后在过程中用 ReDim 语句重新定义数组的大小。即：

```
Public | Private | Dim | Static 数组名 ( ) [As <类型>]
...
ReDim [Preserve] 数组名 (下标1, 下标2, ...) [As 数据类型]
```

说明：

(1) ReDim 语句只能出现在过程中，否则会出现错误。

(2) 静态数组中对应数组下标上界和下界的声明必须使用常量，而用 ReDim 语句定义动态数组时可以使用常量，也可以是有了确定值的变量。例如：

```
Dim a() As Integer, n As Integer
n = 3
ReDim a(n)
```

(3) 在过程中可以多次使用 **ReDim** 语句改变数组的大小, 也可以改变数组的维数, 但不能改变数组的数据类型。例如:

```
Dim a() As Integer
ReDim a(3) ' 定义了一个动态的一维数组
ReDim a(3, 4) As Integer ' 定义了一个动态的二维数组
```

是正确的, 可以多次使用 **ReDim** 改变数组的大小及维数。而下面的代码:

```
Dim a() As Integer
ReDim a(3) As String
```

是错误的, 不能改变数组的数据类型。

(4) 在使用 **ReDim** 语句重新定义数组的大小时, 数组中的数据都会丢失。如果要在使用 **ReDim** 语句重新定义数组的大小时保留数组中的数据, 则必须在 **ReDim** 语句后加 **Preserve** 关键字。使用了 **Preserve** 关键字后将不能再改变数组的维数, 同时也只能改变最后一维的大小, 对于其他的维数不能做任何修改。例如:

```
Dim a() As Integer
ReDim a(3, 4)
ReDim Preserve a(3, 6) As Integer
```

是正确的。而

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

是错误的。使用 **Preserve** 后不能改变数组的维数。

例如多次使用 **ReDim** 语句。

在窗体层声明以下数组:

```
Option Base 1 ' 规定数组默认下界值为1
Private Sub Form_Click()
    Dim sName() As String ' sName为动态字符串类型数组
    ReDim sName(2) ' 第一次重定义含两个元素
    sName(1) = "张朋": sName(2) = "郎悦竹"
    Print sName(1); sName(2)
    ReDim Preserve sName(3) ' 第二次重定义含3个元素, 使用Preserve选项
    sName(3) = "刘凯强"
    Print sName(1); sName(2); sName(3)
    ReDim sName(4) ' 第三次重定义含4个元素, 不使用Preserve选项
    sName(4) = "祁帅"
```

```
Print sName(1); sName(2); sName(3); sName(4)
End Sub
```

上述程序的运行结果如图 5.9 所示，在第二次重定义中使用了 Preserve 选项，因此在第二行可以看到原有的数据元素内容不变。由于在第三次重定义 sName 数组时没有使用 Preserve 选项，sName 数组原先已有的数据都丢失，只有 sName(4) 数组元素有赋值语句，所以第三行的输出结果只有 sName(4) 数组元素内容。

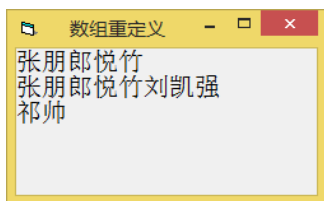


图 5.9 多次使用 ReDim 语句

5.3.2 动态数组的应用

【例 5.7】 由键盘输入 N 名学生的成绩，求出最高分和最低分。

解题思路：在例 5.1 中是求 10 名学生的最高分和最低分，使用了固定数组；本例要求 N 名学生的最高分和最低分，由于 N 未指定，因此应该使用动态数组，将学生的成绩声明为动态数组，程序运行后从键盘上输入 N 的值及学生的成绩。

创建应用程序的用户界面和设置对象属性。在窗体上建立 4 个标签、4 个文本框和一个命令按钮。标签 Label1 的 Caption 属性为“请输入学生人数：”，Label2 的 Caption 属性为“最高分：”，Label3 的 Caption 属性为“最低分：”，Label4 的 Caption 属性为“成绩分别为：”。4 个文本框的名称分别为 Text1、Text2、Text3 和 Text4，Text 属性均为空，Text4 的 MultiLine 属性为 True。命令按钮的名称为 Command1，其 Caption 属性为“判断”。

程序运行后在文本框 Text1 内输入学生人数，单击“判断”按钮后通过键盘依次输入成绩，开始查找最高分和最低分，找到后分别显示在文本框 Text2 和 Text3 中，并将输入的 N 名同学的成绩显示在 Text4 中。程序运行结果如图 5.10 所示。

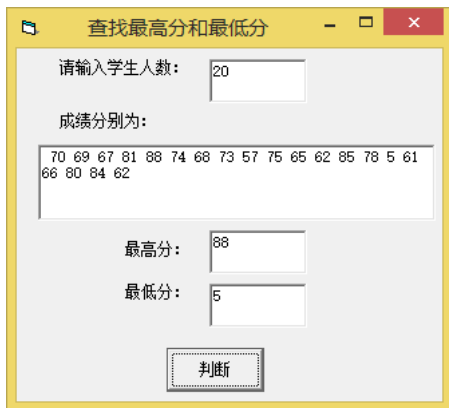


图 5.10 例 5.7 的运行结果

程序代码如下：

```
Private Sub Command1_Click()
    Dim a() As Integer, i As Integer
    Dim n As Integer
    Dim max As Integer, min As Integer
    n = Val(Text1.Text)
    ReDim a(n)
    For i = 1 To n
        a(i) = Val(InputBox("请依次输入学生成绩", "输入成绩"))
    Next i
    For i = 1 To n
        Text4.Text = Text4.Text + Str(a(i))
    Next i
    max = a(1)
    min = a(1)
    For i = 2 To n
        If max < a(i) Then max = a(i)
        If min > a(i) Then min = a(i)
    Next i
    Text2.Text = max
    Text3.Text = min
End Sub
```

请思考：如果学生数为 n 人，课程为 m 门，应该如何编程实现？

【例 5.8】 定义动态数组存放书名并输出，如图 5.11 所示。

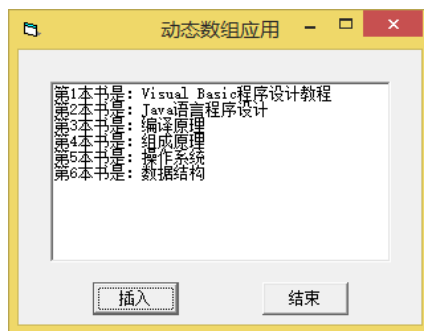


图 5.11 动态数组存放书名

解题思路：定义一个动态数组 `book()`，其大小由循环来控制，每输入一本书名，数组即增加一个元素，直到不输为止。

程序代码如下：

```
Private Sub Command1_Click()
    Text1 = ""
    Dim n As Long, i%
```

```

Dim book()
Do
    ReDim Preserve book(n)      ' 重新定义动态数组book
    book(n) = InputBox("请输入书名,否则结束输入", "提示")
    n = n + 1
Loop Until book(n - 1) = ""
For i = 0 To n - 2
    Text1 = Text1 & "第" & i + 1 & "本书是: " & book(i) & vbCrLf
Next i
End Sub

```

5.4 数组的函数与语句

数组相关函数与语句是数组比较常用的实现方式，本节来认识和学习一些在数组中比较常用的数组相关函数与语句。

1. Array 函数

数组一经定义以后就可以使用 **Array** 函数给数组元素赋初值。**Array** 函数的语法格式如下：

数组变量名 = Array (数组元素值)

功能：把一组数据赋给数组中的每个元素。

说明：

(1) 数组变量名：在使用 **Array** 之前已定义过的数组名，但在定义该数组时像定义变量一样，既没有维数也没有上、下界，并且数组类型必须是 **Variant** 类型。在“数组变量名”之后没有括号。

定义数组变量的方式有下面 3 种：

① 显式定义为 **Variant** 类型。例如：

```
Dim a As Variant
```

② 因为 **Visual Basic** 的默认数据类型就是 **Variant**，所以在定义时可以不指明类型。例如：

```
Dim b
```

③ 如果程序中没有 **Option Explicit** 语句（强制变量声明），可以不定义数组变量，直接使用。

(2) 数组元素值：一个用逗号分隔的值表，这些值构成数组的各元素值。赋值后的数组大小由参数的个数决定。例如：

```

Dim A As Variant
A = Array(10,20,30,40,50)

```

数组 **A** 含 5 个元素。在系统默认情况下数组元素的下标从 0 开始，所以第一个元素

$A(0)$ 的值是"10", 第二个元素 $A(1)$ 的值是"20", 依此类推。

(3) 数组中每个元素的数据类型可以不同。例如:

```
Dim a
a = Array(111, "abc", 2.22) '数组中的各元素赋予不同类型的初值
```

(4) Array 函数只适用于一维数组, 不能对二维或多维数组初始化。

(5) 使用 Array 函数创建的数组, 其下界受 Option Base 语句指定的下界的限制。

【例 5.9】 将一些数值赋给一个动态数组, 然后将其中的偶数显示在窗体上, 已知数据为 “47, -55, 13, 45, 64, 40, 18, 716, -3”。

程序代码如下:

```
Private Sub Command1_Click()
    Dim a , i As Integer
    a=array(47,-55,13,45,64,40,18,716,-3)
    print
    For i = 0 To 8
        If a(i) mod 2=0 then print a(i);spc(3);
    Next i
End Sub
```

运行结果如图 5.12 所示。



图 5.12 例 5.9 的运行结果

2. UBound 函数和 LBound 函数

使用 UBound 函数和 LBound 函数可以求出数组下标的上界和下界。其语法格式为:

UBound (数组名 [,N])

LBound (数组名 [,N])

功能: UBound 函数返回 “数组名” 指定的数组的第 N 维的上界。

LBound 函数返回 “数组名” 指定的数组的第 N 维的下界。

说明: N 为 1 表示第一维, N 为 2 表示第二维, 等等。如果省略 N , 则默认为 1。

例如:

```
Dim a(1 To 5, 2 To 6, 5 To 10) As Integer
```

```
Print LBound(a), UBound(a), LBound(a, 2), UBound(a, 2)
```

程序运行结果为：

```
1 5 2 6
```

【例 5.10】 使用 UBound 函数和 LBound 函数。

代码如下：

```
Private Sub Form_Click()  
    Dim a(10), b(10, 8, -4 To 5) As Integer  
    Print LBound(a), UBound(a)  
    Print  
    Print LBound(b, 1), UBound(b, 1)  
    Print LBound(b, 2), UBound(b, 2)  
    Print LBound(b, 3), UBound(b, 3)  
End Sub
```

程序运行结果如图 5.13 所示。

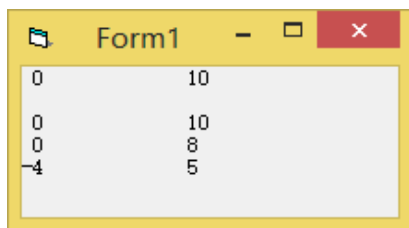


图 5.13 UBound 函数和 LBound 函数示例

3. Split 函数

Split 函数可以从一个字符串中以某个指定符号为分隔符分离若干个子字符串，建立一个下标从零开始的一维数组。Split 函数的使用语法格式如下：

split(<字符串表达式>[, <分隔符>])

功能：以某个指定符号作为分隔符，将“字符串表达式”指定的字符串分离为若干个子字符串，以这些子字符串为元素构成一个下标从零开始的一维数组。

说明：

(1) <字符串表达式>是一个必选项，包含子字符串和分隔符的字符串表达式。如果<字符串表达式>是一个长度为零的字符串（""），Split 返回一个空数组，即没有元素和数据的数组。

(2) <分隔符>是可选的，如果忽略，则使用空格字符（" "）作为分隔符。如果<分隔符>是一个长度为零的字符串，则返回的数组仅包含一个元素，即完整的<字符串表达式>字符串。

例如：

```
Dim A() As String
A = Split("how are you", " ")
```

执行以上赋值之后, A(0)=how, A(1)=are, A(2)=you。

4. Erase 语句

在实际应用中有时需要清除数组的内容或对数组重新定义, 可以用 Erase 语句来实现。其语法格式如下:

Erase <数组名>[, <数组名>]...

功能: 重新初始化静态数组元素, 或者释放动态数组的存储空间。

说明:

(1) 在 Erase 语句中只需给出数组名, 不带括号和范围。

(2) 当 Erase 语句用于静态数组时数组仍然存在, 只是其内容被清空。所有数值型元素置为 0; 所有字符型元素置为空串; 所有 Variant 型元素置为 Empty。

(3) 当 Erase 语句用于动态数组时将删除数组结构并释放数组所占用的内存。在下次引用该动态数组之前必须用 ReDim 语句重新定义该数组。

【例 5.11】 Erase 语句应用举例。

代码如下:

```
Private Sub Form_Click()
    Dim a%(2), b()
    a(0) = 1
    a(1) = 2
    a(2) = 3
    b = Array(4, 5)
    Erase a, b           ' 数组清除
    Print a(0), a(1), a(2)
    ReDim b(3)           ' 动态数组重新定义
    b(0) = 1
    b(1) = 2
    b(2) = 3
    Print b(0), b(1), b(2)
End Sub
```

程序运行结果如图 5.14 所示。

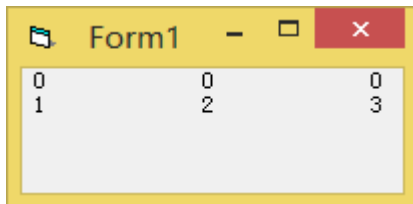


图 5.14 Erase 语句示例

5. For Each...Next 语句

For Each...Next 语句和 For...Next 语句类似，都是实现循环结构的语句，但 For Each...Next 语句是专用于数组和对象集合的。其语法格式如下：

```
For Each 成员 In 数组
    循环体
[ Exit For ]
...
Next 成员
```

(1) “成员”必须是一个变体类型的变量，它是为循环提供的，并在 For Each...Next 语句中重复使用，它实际上代表的是数组中的每个元素。

(2) 数组：在此之前经过定义的数组，用在此处仅仅是一个数组名，没有括号和上、下界。

(3) 循环执行的次数取决于数组元素的个数，数组元素的个数就是循环执行的次数。

(4) 可以使用 Exit For 语句退出循环。

使用 For Each...Next 语句可以对数组元素进行处理，包括查询、显示或读取。

【例 5.12】 统计成绩高于 80 分的学生人数。

解题思路：假设有 N 个学生，学生成绩已知，则定义一个数组变量 Score，用 Array 函数给数组元素赋初值。在 For Each...Next 语句内用 If 语句检查每一个元素的值，如果大于 80 就进行统计（设 Count 为统计人数的变量）。

代码如下：

```
Option Base 1
Private Sub Form_Click()
    Const N = 10
    Dim Count As Integer          ' Count变量用于统计人数
    Dim one, Score                ' one和Score都是Variant类型
    Score = Array(76, 87, 54, 91, 77, 63, 93, 79, 56, 88) ' 数组初始化
    Count = 0
    For Each one In Score
        If one > 80 Then Count = Count + 1 ' 若有成绩>80的数,Count值增1
    Next one
    ' 输出部分
    Print "成绩分别是: "
    For Each one In Score
        Print one;                ' 输出每个元素
    Next one
    Print                          ' 输出换行
    Print "成绩>80 的人数有"; Count; "个"
End Sub
```

程序运行结果如图 5.15 所示。

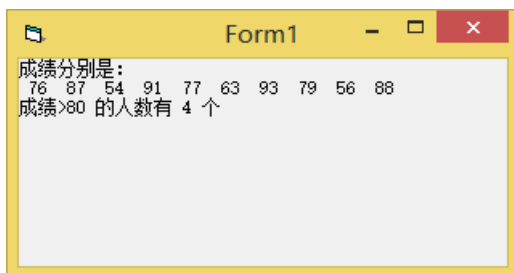


图 5.15 例 5.12 的运行结果

当需要访问数组的所有元素时，使用 For Each...Next 语句比使用 For...Next 语句更加方便，不需要知道数组的上、下界就能对数组的所有元素进行统一的操作，而用 For...Next 语句时必须用一个循环控制变量，要给该变量赋初值，并给出循环结束的条件以及循环变量每次变化的步长值（默认为 1）。

5.5 控件数组

在程序设计中，特别是设计图形界面时，一般将类型相同、功能相似的控件视为一个数组，其使用和普通数组基本相同。

5.5.1 控件数组的定义

一组相同类型的相关数据可以使用数组来描述和管理，在 VB 中一些功能相似的同类控件也可以使用数组进行组织管理，这样的数组称为控件数组。

控件数组是由一组相同类型的控件组成的，它们共用一个控件名。控件数组适用于若干个控件执行的操作相似的场合，控件数组共享同样的事件过程。控件数组通过索引号（属性中的 Index）来标识各控件，第一个下标是 0。例如 Command1(0)、Command1(1)、Command1(2)、Command1(3)……控件数组的下标下界不受语句 Option Base 1 的影响，总是从 0 开始。

5.5.2 控件数组的特点

控件数组中任意一个控件的事件都将触发整个控件数组的事件，不再作为单独控件的事件处理。也就是说，如果建立了一组选项按钮的控件数组，那么无论单击哪个选项按钮都将触发整个控件数组的 Click 事件。为了区分是控件数组中哪个控件产生的事件，VB 将产生事件控件的索引号（Index）传递给控件数组的事件过程。例如，普通选项按钮的 Click 事件过程如下：

```
Private Sub Option1_Click()  
...  
End Sub
```

而一组选项按钮构成的控件数组的 Click 事件过程增加了一个 Index 参数：

```
Private Sub Option1_Click(Index As Integer)
...
End Sub
```

在这里通过 **Index** 参数判断是哪一個选项按钮触发的 Click 事件, 在建立控件数组之后, 无论单击哪一个单选按钮都将触发这个 Click 事件过程。

5.5.3 控件数组的创建

创建控件数组主要有复制粘贴、设置控件的 **Name** 属性和在程序中定义 3 种方法。

1. 复制粘贴法

通过复制粘贴控件创建控件数组, 具体步骤如下:

- (1) 在窗体上添加一个要创建控件数组的控件。
- (2) 选中该控件右击, 在弹出的菜单中选择“复制”命令。
- (3) 使用鼠标选中窗体, 然后右击, 在弹出的菜单中选择“粘贴”命令, 此时会弹出一个如图 5.16 所示的提示对话框。单击“是”按钮, 则在窗体上添加一个新的控件数组元素。



图 5.16 创建控件数组

- (4) 重复执行步骤 (3), 直到添加完所需的控件数组元素为止。

2. 设置控件的 **Name** 属性

控件的 **Name** 属性用来标识控件的名字, 将需要放置在控件数组中的同类型控件的 **Name** 属性设置为相同名称也可以创建控件数组, 具体步骤如下:

- (1) 在窗体或容器控件中添加两个或多个同类型的要创建控件数组的控件。
- (2) 逐一选中添加的每个控件, 并设置相同的 **Name**, 完成控件数组的创建。设置 **Name** 属性, 当第一次出现同名 **Name** 时也会出现如图 5.16 所示的提示对话框, 单击“是”按钮即可创建控件数组。后续控件的 **Name** 属性设置不再出现提示对话框。

3. 程序中 **Load** 方法

在程序运行时通过 **Load** 方法完成控件元素相同属性的设置, 实现控件数组的创建, 具体步骤如下:

- (1) 在窗体或容器中添加一个控件, 并设置 **Index** 属性为 0, 表示这是一个控件数组 (一个元素)。
- (2) 在代码中通过 **Load** 方法添加其余控件元素, 用 **Unload** 方法删除其中的控件元素。

注意: 用此方法添加的控件元素还需进行 **Top**、**Left** 属性的设置, 并将 **Visible** 属性设置为 **True**, 这样才能使添加的控件元素显示在窗体上。

5.5.4 控件数组的应用

【例 5.13】建立含有 4 个命令按钮的控件数组，当单击某个命令按钮时分别执行不同的操作。

步骤如下：

- (1) 在窗体上建立一个命令按钮，然后用“复制粘贴法”创建另外 3 个命令按钮。
- (2) 把 4 个命令按钮的 Caption 属性分别设置为“命令按钮 1”、“命令按钮 2”、“命令按钮 3”和“命令按钮 4”。
- (3) 双击任意一个命令按钮打开代码窗口，输入以下事件过程：

```
Private Sub Command1_Click(Index As Integer)
    FontSize = 12
    If Index = 0 Then
        Print "单击了第一个命令按钮"
    ElseIf Index = 1 Then
        Print "单击了第二个命令按钮"
    ElseIf Index = 2 Then
        Print "单击了第三个命令按钮"
    Else
        Print "单击了第四个命令按钮"
    End If
End Sub
```

上述过程根据 Index 属性的值决定在单击某个命令按钮时所执行的操作。所建立的控件数组包括 4 个命令按钮，其下标（Index 属性）分别为 0、1、2、3。第一个命令按钮的 Index 属性为 0，因此，当单击第一个命令按钮时执行的是下标为 0 的那个数组元素的操作；而当单击第二个命令按钮时执行的是下标为 1 的那个数组元素的操作，等等。程序运行结果如图 5.17 所示。

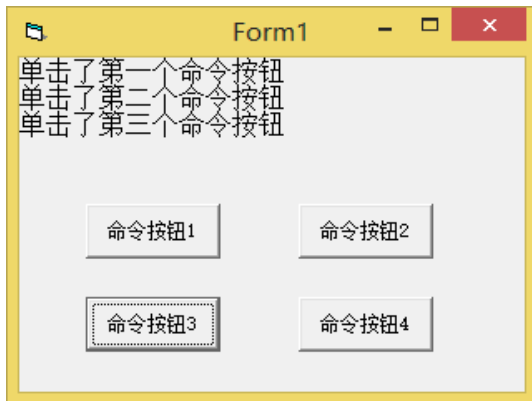


图 5.17 例 5.13 的运行结果

【例 5.14】 设计窗体，输入两个数，根据不同运算符计算结果。

设计步骤如下：

建立应用程序用户界面并设置对象属性。在窗体上添加 3 个标签对象 Label1~Label3；3 个文本框对象 Text1~Text3，其 Text 属性均为空，并且将 Text3 的 Enabled 属性设置为 False；一个框架对象 Frame1，并在 Frame1 内添加选项按钮控件数组。界面设计如图 5.18 所示。

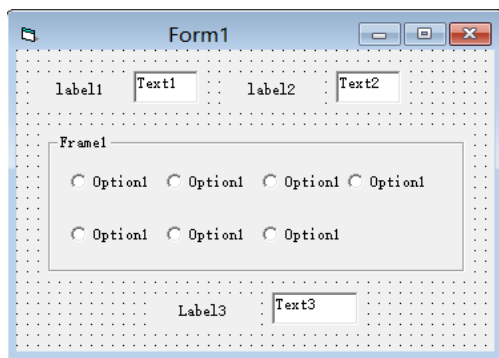


图 5.18 例 5.14 的界面设计

代码如下：

```
Private Sub Option1_Click(Index As Integer)
    Dim x As Single, y As Single
    x = Val(Text1.Text)
    y = Val(Text2.Text)
    Select Case Index          ' Index值从0开始
        Case 0                ' 加
            Text3.Text = x + y
        Case 1                ' 减
            Text3.Text = x - y
        Case 2                ' 乘
            Text3.Text = x * y
        Case 3                ' 除
            Text3.Text = x / y
        Case 4                ' 整除
            Text3.Text = x \ y
        Case 5                ' 取余
            Text3.Text = x Mod y
        Case 6                ' 指数
            Text3.Text = x ^ y
    End Select
End Sub
```

程序运行结果如图 5.19 所示。



图 5.19 例 5.14 的运行结果

习 题 5

1. 简答题

- (1) 什么是数组？
- (2) 静态数组与动态数组的主要区别是什么？
- (3) 如何创建控件数组？
- (4) 简述数组的相关操作语句和函数的基本含义及作用。
- (5) 使用 Array 函数进行数组初始化时需要注意的问题是什么？

2. 填空题

- (1) 在 VB 6.0 中可以通过 () 语句强制数组下标从 1 开始。
- (2) 如果定义一个数组 “Dim A(6) As Integer”，其元素最多有 () 个，如果之前在窗体中放置了语句 “Option Base 1”，则元素的个数有 () 个。
- (3) 控件数组的名字由 () 属性指定，而数组中的每个元素由 () 属性指定。
- (4) 以下程序段运行后的输出结果是 ()。

```
Dim a(6)
For i = 1 To 6
    a(i) = i * i
Next i
Print a(i - 1)
```

- (5) 以下程序段运行后的输出结果是 ()。

```
Dim a(100)
For i = 1 To 100
    a(i) = 2 * i
Next i
Print a(a(22) - 1)
```

- (6) 窗体上有从左到右 4 个选项按钮组成的控件数组 Option1，下标从 0 开始。程序运行时，单击“选择”命令按钮（名称为 Command1），则在标签 Label1 中显示所选中的

信息，如图 5.20 所示。

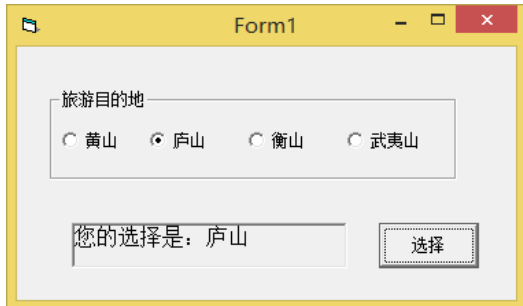


图 5.20 填空题（6）的图

以下是完成上述功能的程序，请填空。

```
Private Sub Command1_Click()  
    For i = 0 To 3  
        If Option1(i).Value = (    ) Then  
            Label1.Caption = "您的选择是：" & (    ).Caption  
        End If  
    Next i  
End Sub
```

（7）窗体上有名称为 **Command1** 的命令按钮，其事件过程如下：

```
Option Base 1  
Private Sub Command1_Click()  
    Dim a  
    Dim i As Integer, n As Integer, s As Integer  
    a = Array(9, 8, 7)  
    n = 1  
    For i = 1 To 3  
        s = s + a(i) * n  
        n = n * 10  
    Next i  
    Print s  
End Sub
```

运行程序，单击命令按钮，窗体上显示的内容是（ ）。

（8）以下程序的功能是从随机产生的 20 个 20~200（含 20 和 200）的整数中找出能够同时被 3 和 5 整除的数的最大值，并显示出来，请填空。

```
Option Base 1  
Private Sub Command1_Click()  
    Dim a(20) As Integer  
    m = 0  
    For i = 1 To 20
```

```

a(i) = Int(Rnd * _____) + 20
If a(i) Mod 3 = 0 _____ a(i) Mod 5 = 0 Then
    If a(i) > m Then m = a(i)
End If
Next i
If _____ Then
    Print "没有满足条件的数"
Else
    Print "满足条件的数是"; m
End If
End Sub

```

3. 选择题

(1) 用 Dim B%(5)语句声明的数组的全部元素都初始化为 ()。

- A. 0 B. 1 C. False D. 空字符串

(2) 以下有关数组定义的语句序列中错误的是 ()。

- A. Static arr1(3)

```

arr1(1)=100
arr1(2)="Hello"
arr1(3)=123.45

```

- B. Option Base 1

```

Private Sub Command3_Click()
    Dim arr3(3) As Integer
    ...
End Sub

```

- C. Dim arr2() As Integer

```

Dim size As Integer
Private Sub Command2_Click()
    size=InputBox("输入: ")
    ReDim arr2(size)
    ...
End Sub

```

- D. Dim n As Integer

```

Private Sub Command4_Click()
    Dim arr4(n) As Integer
    ...
End Sub

```

(3) 用下面的语句定义的数组元素的个数是 ()。

```
Dim Array(3 To 5,-2 To 2) As Integer
```

- A. 20 B. 12 C. 15 D. 24

(4) 以下说法不正确的是 ()。

- A. 使用 ReDim 语句可以改变数组的维数
- B. 使用 ReDim 语句可以改变数组的类型
- C. 使用 ReDim 语句可以改变数组的每一维的大小
- D. 使用 ReDim 语句可以对数组中的所有元素进行初始化

(5) 使用复制、粘贴的方法建立了一个命令按钮数组 Command1, 以下对该数组的说法错误的是 ()。

- A. 命令按钮的所有 Caption 属性都是 Command1
- B. 在代码中访问命令按钮只需使用名称 Command1
- C. 命令按钮的大小都相同
- D. 命令按钮共享相同的事件过程

(6) 设执行以下程序段时依次输入 1、3、5, 执行结果为 ()。

```
Dim a(4) As Integer, b(4) As Integer
For K = 0 To 2
    a(K + 1) = Val(InputBox("请输入数据"))
    b(3 - K) = a(K + 1)
Next K
Print b(K)
```

- A. 0 B. 3 C. 5 D. 1

(7) 以下程序段的执行结果为 ()。

```
Dim a(-2 To 5)
For i = LBound(a, 1) To UBound(a, 1)
    a(i) = i * i
Next i
Print a(LBound(a, 1)); a(UBound(a, 1))
```

- A. 0 0 B. -2 5 C. 4 25 D. 1 25

(8) 设有以下程序:

```
Option Base 0
Private Sub Form_Click()
    Dim a
    Dim i As Integer
    a = Array(1, 2, 3, 4, 5, 6, 7, 8, 9)
    For i = 0 To 3
        Print a(5 - i);
    Next i
End Sub
```

程序运行后单击窗体, 则在窗体上显示的是 ()。

- A. 6 5 4 3 B. 5 4 3 2 C. 4 3 2 1 D. 7 6 5 4

(9) 窗体上有名称为 Command1 的命令按钮, 其事件过程如下:

```
Private Sub Command1_Click()  
    Static x As Variant  
    n = 0  
    x = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)  
    While n <= 4  
        x(n) = x(n + 5)  
        Print x(n);  
        n = n + 1  
    Wend  
End Sub
```

运行程序, 单击命令按钮, 窗体上显示的是 ()。

- A. 1 2 3 4 5 B. 6 7 8 9 10 C. 2 3 4 5 6 D. 6 2 3 4 5

(10) 窗体上有名称为 Command1 的命令按钮, 其事件过程如下:

```
Private Sub Command1_Click()  
    Dim a, b(2, 3) As Integer  
    a = Array(3, 7, 5, 11, 31, 43, 17, 62, 9, 23, 37, 41)  
    i = 0  
    Do While i <= UBound(a)  
        For j = 0 To 2  
            For k = 0 To 3  
                b(j, k) = a(i)  
                i = i + 1  
            Next k  
        Next j  
    Loop  
    Print b(2, 2)  
End Sub
```

运行程序, 单击命令按钮, 窗体上显示的是 ()。

- A. 43 B. 17 C. 37 D. 23

(11) 在窗体上画一个名称为 Command1 的命令按钮, 然后编写以下程序:

```
Private Sub Command1_Click()  
    Dim i As Integer, j As Integer  
    Dim a(10, 10) As Integer  
    For i=1 To 3  
        For j=1 To 3  
            a(i, j)=(i-1)*3+j  
            Print a(i, j);  
        Next j  
        Print  
    Next i  
End Sub
```

程序运行后，单击命令按钮，窗体上显示的是（ ）。

- | | | | | | | | |
|----|-------|----|-------|----|-------|----|-------|
| A. | 1 2 3 | B. | 2 3 4 | C. | 1 4 7 | D. | 1 2 3 |
| | 4 5 6 | | 3 4 5 | | 2 5 8 | | 2 4 6 |
| | 7 8 9 | | 4 5 6 | | 3 6 9 | | 3 6 9 |

4. 编程题

(1) 从键盘上输入 10 个数，并放入一个一维数组中，然后将其前 5 个元素与后 5 个元素互换，即第 1 个元素与第 10 个元素互换，第 2 个元素与第 9 个元素互换……第 5 个元素与第 6 个元素互换，分别输出原来各元素的值和对换后元素的值。

(2) 找出二维数组 $m \times n$ 中的“鞍点”，输出该点的位置和值。若不存在鞍点，则输出“没有鞍点”的提示信息。所谓“鞍点”是指该点的数据在所在行最大，在所在列最小。