Program Example 17   Variable Arrays 1

Objectives

**Program Purpose**

- To generate, store and recall a random pattern of six coloured shapes

**Learning Goals**

- To use a Control Array with a Variable Array
- To understand the differences between Control and Variable Arrays
- To understand how to create, copy and paste controls in frames

Design Notes

This project demonstrates how control arrays and variable arrays can be used together.

Set the properties for one label, then copy and paste it. When pasting, create a control array.

Always set the name properties before entering any code. Labels that serve no other purpose than displaying text do not need particular names. The default names Label1, Label2 etc. are sufficient.

Interface

Create the interface as shown.

Use 1 form, 2 separate arrays of 6 shapes each (pasted into 2 separate frames) and 4 command buttons.

Be very careful when copying and pasting the shapes in the frames. Create the first frame, put the first shape into that frame and then copy and paste the remaining shapes while the frame is selected. Answer 'Yes' to the Control Array question. When creating the second frame, do not copy and paste a shape from the first frame. Instead, draw the first shape into the new frame and then copy and paste from this.

Names of Objects

<table>
<thead>
<tr>
<th>Type of Object</th>
<th>Number</th>
<th>Names of Objects</th>
<th>Simple Initial Properties of Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form</td>
<td>1</td>
<td>Form1</td>
<td>Caption - &quot;Variable Arrays 1&quot; Icon - Browse and choose any icon Startup Position - Centre Screen</td>
</tr>
<tr>
<td>Frames</td>
<td>2</td>
<td>No name necessary</td>
<td>Captions - Pattern1, Pattern 2 Font - Bold,12</td>
</tr>
<tr>
<td>Control arrays of shapes</td>
<td>2</td>
<td>shpP1(0-5), shpP2(0-5)</td>
<td>BackStyle - Opaque BackColor - White Shape - 0 (Rectangle)</td>
</tr>
<tr>
<td>Command Buttons</td>
<td>4</td>
<td>cmdGenerate, cmdStore, cmdRecall, cmdClear</td>
<td>Font - Bold, 12 Captions - as per interface</td>
</tr>
</tbody>
</table>
GENERAL SECTION

Dim z As Integer
Const NumCells = 6
Dim Pattern1(NumCells-1) As Long

Private Sub cmdClear_Click()
    For z = 0 To NumCells - 1
        shpP1(z).BackColor = vbWhite
        shpP2(z).BackColor = vbWhite
    Next z
End Sub

Private Sub cmdGenerate_Click()
    Randomize
    For z = 0 To NumCells - 1
        shpP1(z).BackColor = RGB(Rnd*255, Rnd*255, Rnd*255)
    Next z
End Sub

Private Sub cmdRecall_Click()
    For z = 0 To NumCells - 1
        shpP2(z).BackColor = Pattern1(z)
    Next z
End Sub

Private Sub cmdStore_Click()
    For z = 0 To NumCells - 1
        Pattern1(z) = shpP1(z).BackColor
    Next z
End Sub

Consolidation and Extension

1. Which lines of code initialise the two arrays as white?
2. Which lines of code set the random back colours?
3. Which lines of code store the colours?
4. Which lines of code recall the colours?
5. Double the number of shapes in each array. Make one change to the code and check that the new shapes work.
6. Which line of code declares the Variable Array that stores the colour values of the pattern?
7. The constant NumCells is used in the array declaration, i.e.

   Dim Pattern1(NumCells-1) as Integer.

   This is not necessary, but it is a tidy way to ensure the array is always the correct size when changing the number of shapes. Write down a simpler declaration that would also work in the first instance.
8. Design a new program from scratch. On the interface, create four different patterns. Randomly generate the colours. Store and recall each one separately. Use a separate frame for each of the patterns.