

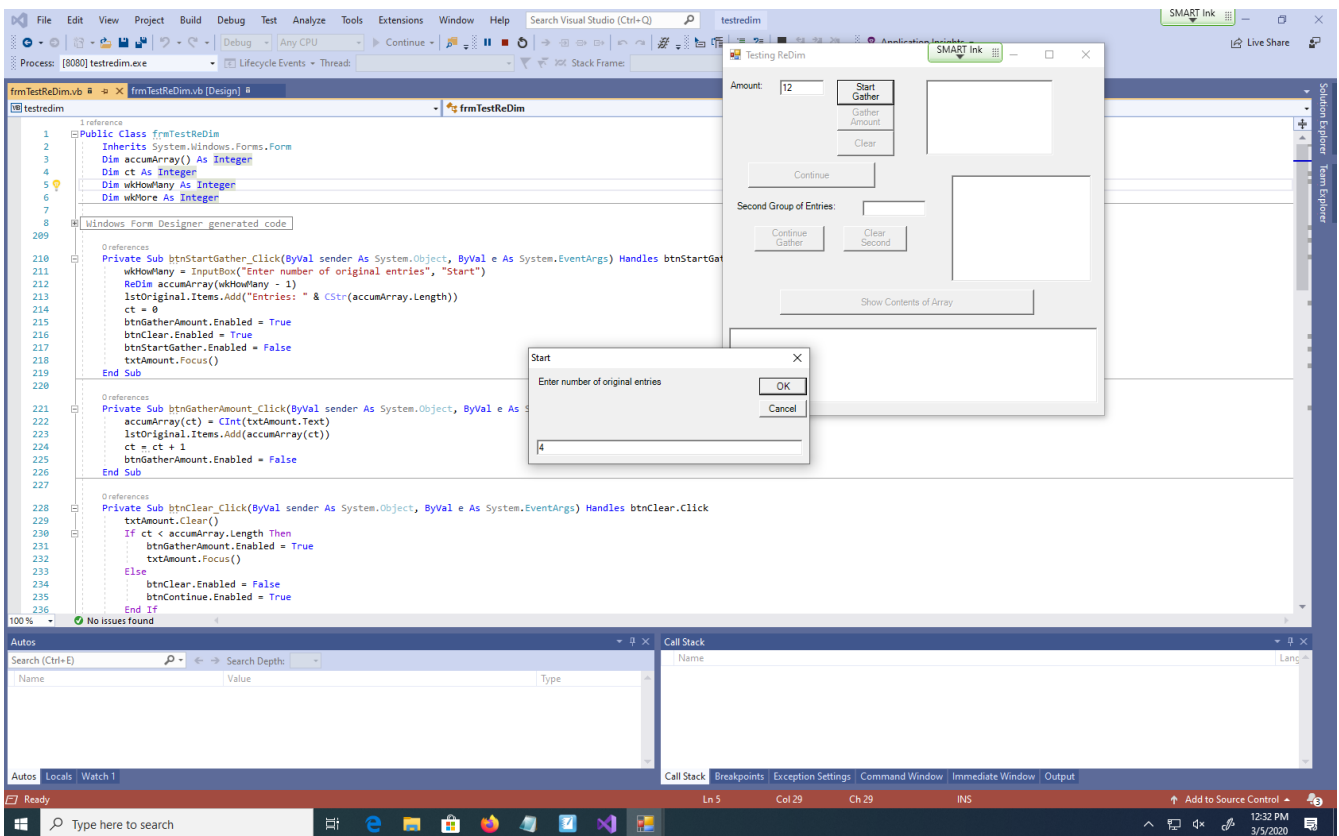
The screenshot displays the Visual Studio IDE with a VB.NET code file open. The code defines a class `frmTestReDim` with several methods for handling array resizing and data collection. A dialog box titled "Testing ReDim" is overlaid on the code, showing a user interface with input fields and buttons. The dialog has two sections: "Amount:" with a text box containing "12" and buttons "Start Gather", "Gather Amount", and "Clear"; and "Second Group of Entries:" with another text box and buttons "Continue Gather" and "Clear Second". A "Show Contents of Array" button is located below the second section. The code in the background includes:

```
Public Class frmTestReDim
    Inherits System.Windows.Forms.Form
    Dim accumArray() As Integer
    Dim ct As Integer
    Dim wkHowMany As Integer
    Dim wkMore As Integer

    Private Sub btnStartGather_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnStartGather.Click
        wkHowMany = InputBox("Enter number of original entries", "Start")
        ReDim accumArray(wkHowMany - 1)
        lstOriginal.Items.Add("Entries: " & CStr(accumArray.Length))
        ct = 0
        btnGatherAmount.Enabled = True
        btnClear.Enabled = True
        btnStartGather.Enabled = False
        txtAmount.Focus()
    End Sub

    Private Sub btnGatherAmount_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGatherAmount.Click
        accumArray(ct) = CInt(txtAmount.Text)
        lstOriginal.Items.Add(accumArray(ct))
        ct = ct + 1
        btnGatherAmount.Enabled = False
    End Sub

    Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClear.Click
        txtAmount.Clear()
        If ct < accumArray.Length Then
            btnGatherAmount.Enabled = True
            txtAmount.Focus()
        Else
            btnClear.Enabled = False
            btnContinue.Enabled = True
        End If
    End Sub
End Class
```



The screenshot displays the Visual Studio IDE with a VB.NET code file open. The code defines three event handlers for a form: `btnStartGather_Click`, `btnGatherAmount_Click`, and `btnClear_Click`. The `btnStartGather_Click` handler initializes an array and a counter. The `btnGatherAmount_Click` handler adds user input to the array. The `btnClear_Click` handler clears the array and enables the 'Gather Amount' button. The `btnContinue_Click` handler adds a second group of entries to the array. A 'Testing ReDim' dialog box is overlaid on the code, showing the current state of the application with a list of entries and buttons for 'Start Gather', 'Gather Amount', 'Clear', 'Continue', 'Continue Gather', 'Clear Second', and 'Show Contents of Array'.

```
210 Private Sub btnStartGather_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnStartGather.Click
211     wkHowMany = InputBox("Enter number of original entries", "Start")
212     ReDim accumArray(wkHowMany - 1)
213     lstOriginal.Items.Add("Entries: " & CStr(accumArray.Length))
214     ct = 0
215     btnGatherAmount.Enabled = True
216     btnClear.Enabled = True
217     btnStartGather.Enabled = False
218     txtAmount.Focus()
219 End Sub
221 Private Sub btnGatherAmount_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGatherAmount.Click
222     accumArray(ct) = CInt(txtAmount.Text)
223     lstOriginal.Items.Add(accumArray(ct))
224     ct = ct + 1
225     btnGatherAmount.Enabled = False
226 End Sub
228 Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClear.Click
229     txtAmount.Clear()
230     If ct < accumArray.Length Then
231         btnGatherAmount.Enabled = True
232         txtAmount.Focus()
233     Else
234         btnClear.Enabled = False
235         btnContinue.Enabled = True
236     End If
237 End Sub
239 Private Sub btnContinue_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnContinue.Click
240     wkMore = InputBox("How many more do you want to enter", "Continue Array")
241     ReDim Preserve accumArray(wkHowMany - 1 + wkMore)
242     lstSecond.Items.Add("Entries: " & CStr(accumArray.Length))
243     btnContinueGather.Enabled = True
```

The screenshot displays the Visual Studio IDE with a VB.NET code file named `frmTestReDim.vb` open. The code implements a program that dynamically resizes an array. It features several buttons: `Start Gather`, `Gather Amount`, `Clear`, `Continue`, `Continue Gather`, `Clear Second`, and `Show Contents of Array`. The code uses `ReDim Preserve` to increase the size of the `accumArray` when the `Continue Array` button is clicked. It also includes logic to manage a second group of entries and to display the contents of the array.

```
235     btnContinue.Enabled = True
236     End If
237 End Sub
238
239 Private Sub btnContinue_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnContinue.Click
240     wkMore = InputBox("How many more do you want to enter", "Continue Array")
241     ReDim Preserve accumArray(wkHowMany - 1 + wkMore)
242     lstSecond.Items.Add("Entries: " & CStr(accumArray.Length))
243     btnContinue.Enabled = True
244     btnClearSecond.Enabled = True
245 End Sub
246
247 Private Sub btnContGather_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnContGather.Click
248     accumArray(ct) = CInt(txtAmountSecond.Text)
249     lstSecond.Items.Add(accumArray(ct))
250     ct = ct + 1
251     btnContGather.Enabled = False
252 End Sub
253
254 Private Sub btnClearSecond_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClearSecond.Click
255     txtAmountSecond.Clear()
256     If ct < accumArray.Length Then
257         btnContGather.Enabled = True
258         txtAmountSecond.Focus()
259     Else
260         btnShowContents.Enabled = True
261     End If
262 End Sub
263
264 Private Sub btnShowContents_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnShowContents.Click
265     Dim i As Integer
266     For i = 0 To accumArray.Length - 1
267         lstShowArray.Items.Add(accumArray(i))
268     Next
269 End Sub
270 End Class
271
```

The application window, titled "Testing ReDim", shows the following state:

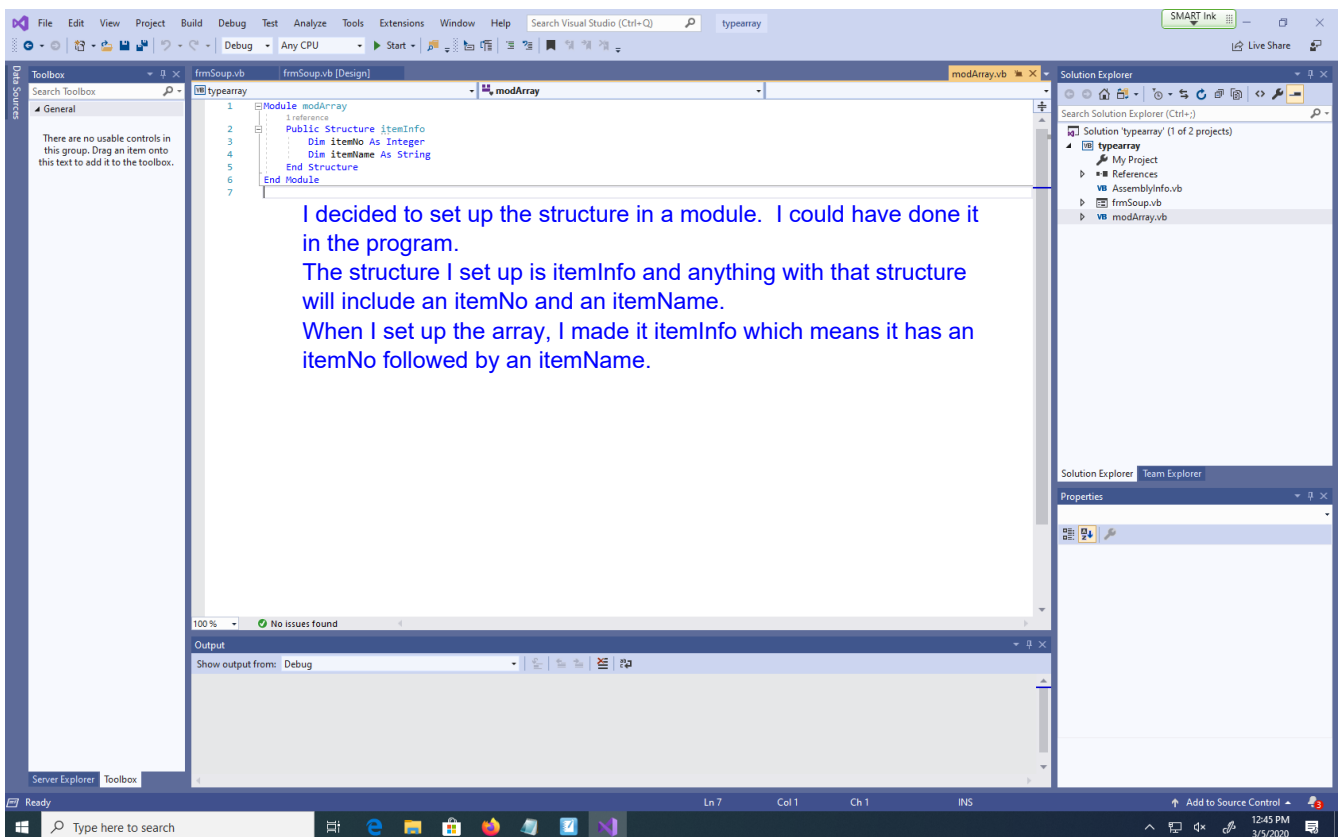
- `Amount:` [Empty text box]
- `Start Gather`, `Gather Amount`, `Clear`, and `Continue` buttons are visible.
- `Entries: 4` is displayed above a list box containing: 12, 15, 17, 27.
- `Continue Gather` and `Clear Second` buttons are visible.
- `Second Group of Entries:` [Empty text box]
- `Entries: 6` is displayed above a list box containing: 36, 52.
- `Show Contents of Array` button is visible.
- A separate list box at the bottom contains: 12, 15, 17, 27, 36, 52.

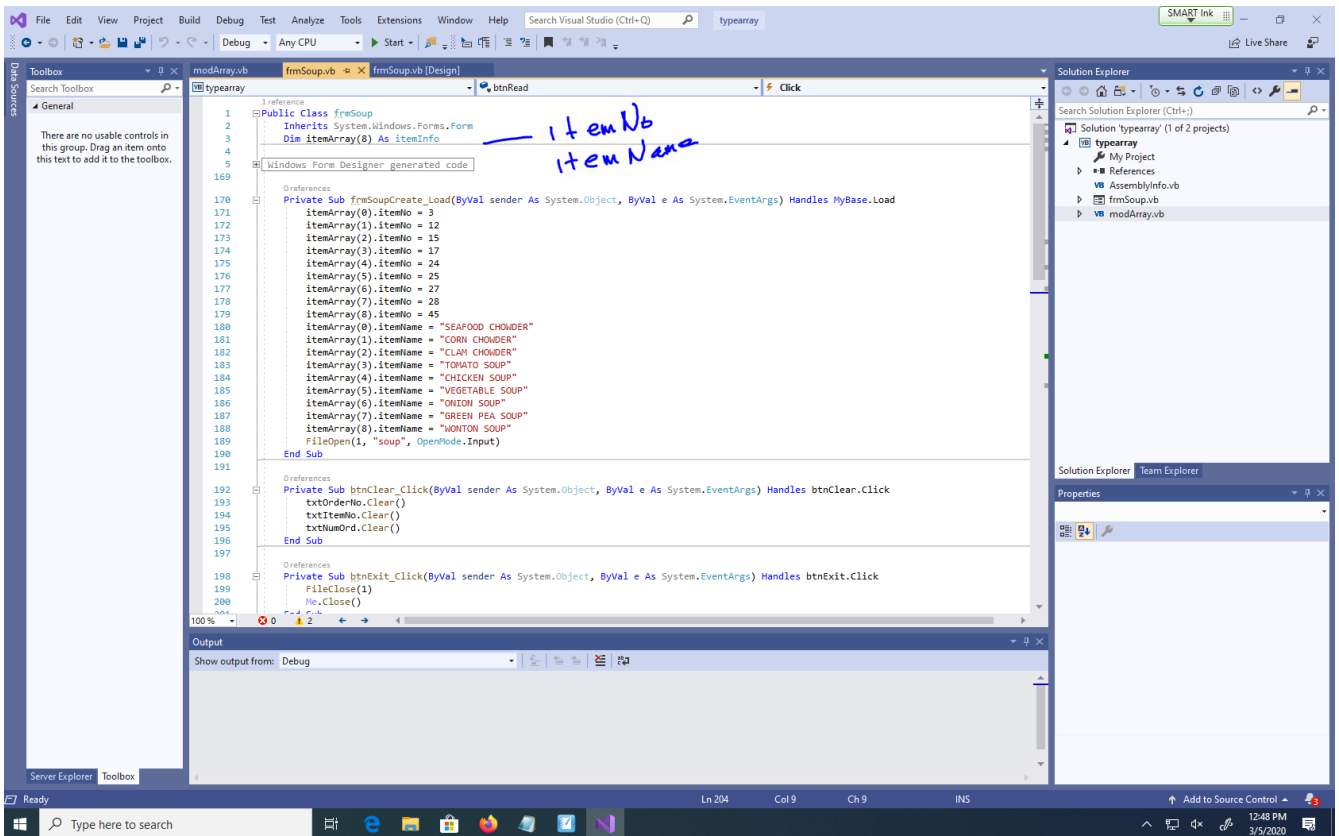
The screenshot displays the Visual Studio IDE with a C# code file named `frmTestReDim.vb` in the Design view. The code implements a program that dynamically resizes an array. It features three main buttons: `Continue`, `Clear Second`, and `Show Contents of Array`. The `Continue` button prompts the user for the number of additional elements to add. The `Clear Second` button clears the second group of entries. The `Show Contents of Array` button displays the current state of the array.

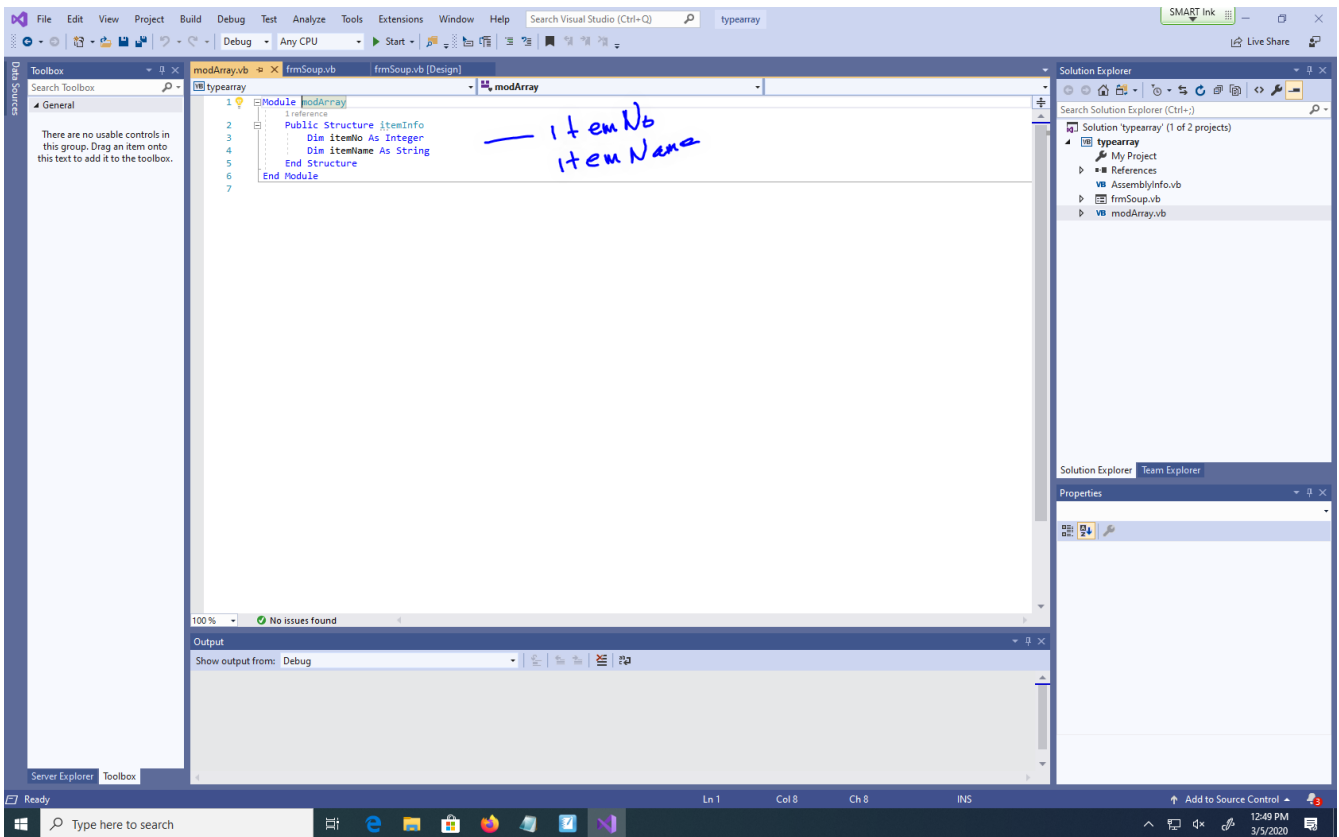
```
235     btnContinue.Enabled = True
236   End If
237 End Sub
238
239 References
240 Private Sub btnContinue_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnContinue.Click
241     wkMore = InputBox("How many more do you want to enter", "Continue Array")
242     ReDim Preserve accumArray(wkHowMany - 1 + wkMore)
243     lstSecond.Items.Add("Entries: " & CStr(accumArray.Length))
244     btnContGather.Enabled = True
245     btnClearSecond.Enabled = True
246 End Sub
247
248 References
249 Private Sub btnContGather_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnContGather.Click
250     accumArray(ct) = CInt(txtAmountSecond.Text)
251     lstSecond.Items.Add(accumArray(ct))
252     ct = ct + 1
253     btnContGather.Enabled = False
254 End Sub
255
256 References
257 Private Sub btnClearSecond_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClearSecond.Click
258     txtAmountSecond.Clear()
259     If ct < accumArray.Length Then
260         btnContGather.Enabled = True
261     Else
262         txtAmountSecond.Focus()
263     End If
264 End Sub
265
266 References
267 Private Sub btnShowContents_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnShowContents.Click
268     Dim i As Integer
269     For i = 0 To accumArray.Length - 1
270         lstShowArray.Items.Add(accumArray(i))
271     Next
272 End Sub
273 End Class
```

The running application window, titled "Testing ReDim", shows the following state:

- `Amount:` [Empty text box]
- `Start Gather`, `Gather Amount`, `Clear` buttons are visible.
- `Continue` button is visible.
- `Second Group of Entries:` [Empty text box]
- `Continue Gather`, `Clear Second` buttons are visible.
- `Show Contents of Array` button is visible.
- `Entries: 4` list box contains: 12, 15, 17, 27.
- `Entries: 6` list box contains: 36, 52.
- Bottom list box contains: 12, 15, 17, 27, 36, 52.







The screenshot shows the Visual Studio IDE with a VB.NET project named 'typearray'. The code in the 'frmSoup.vb' file includes a 'btnRead_Click' event handler and a 'SearchArray' function. The 'btnRead_Click' method takes 'wkOrderNo' and 'wkItemNo' as parameters and uses them to populate a dialog box. The 'SearchArray' function iterates through an array of items to find a match for the provided 'wkItemNo'.

```
197  
198 Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnExit.Click  
199     FileClose(1)  
200     Me.Close()  
201 End Sub  
202  
203  
204 Private Sub btnRead_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnRead.Click  
205     Dim wkOrderNo As Integer, wkItemNo As Integer  
206     Dim wkItemOrd As Integer  
207     If Not EOF(1) Then  
208         Input(1, wkOrderNo)  
209         Input(1, wkItemNo)  
210         Input(1, wkItemOrd)  
211         txtOrderNo.Text = wkOrderNo  
212         txtItemNo.Text = wkItemNo  
213         txtItemOrd.Text = wkItemOrd  
214         txtItemName.Text = SearchArray(wkItemNo)  
215     Else  
216         MessageBox.Show("EOF reached")  
217         btnRead.Visible = False  
218     End If  
219 End Sub  
220  
221 Function SearchArray(ByVal wkItemNo)  
222     Dim itemSub As Integer = 0  
223     Dim matchInd As String = "No"  
224     Do Until itemSub > 8 Or matchInd = "YES"  
225         If wkItemNo = itemArray(itemSub).itemNo Then  
226             matchInd = "YES"  
227         Else  
228             itemSub = itemSub + 1  
229         End If  
230     Loop  
231     If matchInd = "YES" Then  
232         Return itemArray(itemSub).itemName  
233     Else  
234         Return "Match Not Found"  
235     End If  
236 End Function  
End Class
```

Notice that when I use the itemArray I refer to either the itemNo or the itemName since they are both in the array following the structure.

The dialog box 'Search Soup' contains the following data:
Order #: 1212
Item #: 27
Number Ordered: 2
Item Name: ONION SOUP

The screenshot displays the Visual Studio IDE with a VB.NET project named 'typearray'. The code in the 'frmSoup.vb' file includes a 'btnExit_Click' event handler, a 'btnRead_Click' event handler, and a 'SearchArray' function. The 'btnRead_Click' handler prompts the user for an order number, item number, and number of items ordered, then calls 'SearchArray' to find the item name. The 'SearchArray' function iterates through an array of items to find a match. A dialog box titled 'Search So' is overlaid on the code, showing the following fields and values:

Field	Value
Order #:	4078
Item #:	50
Number Ordered:	2
Item Name:	Match Not Found

The dialog box also contains 'Read', 'Clear', and 'Exit' buttons.

```
1 Public Class frmSoup
2 Inherits System.Windows.Forms.Form
3 Dim itemArray(8) As itemInfo
4
5 Windows Form Designer generated code
169
170 Private Sub frmSoupCreate_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
171 itemArray(0).itemNo = 3
172 itemArray(1).itemNo = 12
173 itemArray(2).itemNo = 15
174 itemArray(3).itemNo = 17
175 itemArray(4).itemNo = 24
176 itemArray(5).itemNo = 25
177 itemArray(6).itemNo = 27
178 itemArray(7).itemNo = 28
179 itemArray(8).itemNo = 45
180 itemArray(0).itemName = "SEAFOOD CHOWDER"
181 itemArray(1).itemName = "CORN CHOWDER"
182 itemArray(2).itemName = "CLAM CHOWDER"
183 itemArray(3).itemName = "TOMATO SOUP"
184 itemArray(4).itemName = "CHICKEN SOUP"
185 itemArray(5).itemName = "VEGETABLE SOUP"
186 itemArray(6).itemName = "ONION SOUP"
187 itemArray(7).itemName = "GREEN PEA SOUP"
188 itemArray(8).itemName = "MONTON SOUP"
189 FileOpen(1, "soup", OpenMode.Input)
190 End Sub
191
192 Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClear.Click
193 txtOrderNo.Clear()
194 txtItemNo.Clear()
195 txtNumOrd.Clear()
196 End Sub
197
198 Private Sub btnExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnExit.Click
199 FileClose(1)
200 Me.Close()
201 End Sub
202
203 Private Sub btnRead_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnRead.Click
204 Dim wkOrderNo As Integer, wkItemNo As Integer
205 Dim wkNumOrd As Integer
206 If Not EOF(1) Then
207 Input(1, wkOrderNo)
208 Input(1, wkItemNo)
209 Input(1, wkNumOrd)
210 txtOrderNo.Text = wkOrderNo
```

The screenshot displays the Visual Studio IDE with a VB.NET code file named `frmAirFare.vb` open. The code implements a flight fare calculation application. It includes a `btnGetPrice_Click` event handler that validates input and calculates the fare based on a 2D array of prices. A `frmAirFare_Load` event handler initializes the city lists and price array. A `btnClear_Click` event handler is also present. The application window, titled `AirFare Au`, is running and shows the following state:

- `From Code:` 2
- `To Code:` 2
- Message box: "The price to fly from Austin, TX to London is \$496.98"
- Buttons: `Get Price`, `Clear`, `Exit`, `Get Worthless Total`, `Another`

```
171
172
173 Private Sub btnGetPrice_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGetPrice.Click
174 Dim wkMsg As String
175 If IsNumeric(txtFrom.Text) And CInt(txtFrom.Text) < 4 Then
176 fromPtr = CInt(txtFrom.Text)
177 If IsNumeric(txtTo.Text) And CInt(txtTo.Text) < 3 Then
178 toPtr = CInt(txtTo.Text)
179 wkPrice = priceArray(fromPtr, toPtr)
180 wkMsg = "The price to fly from " & fromCity(fromPtr) & _
181 to " & toCity(toPtr) & " is " & _
182 FormatCurrency(wkPrice)
183 txtFare.Text = wkMsg
184 Else
185 txtFare.Text = "From City is valid, To City is not valid"
186 End If
187 Else
188 If IsNumeric(txtTo.Text) And CInt(txtTo.Text) < 3 Then
189 txtFare.Text = "From City is not valid, To City is valid"
190 Else
191 txtFare.Text = "Both From City and To City are invalid"
192 End If
193 End Sub
194
195
196 Private Sub frmAirFare_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
197 fromCity(0) = "Boston, MA"
198 fromCity(1) = "New York, NY"
199 fromCity(2) = "Austin, TX"
200 fromCity(3) = "Washington, DC"
201 toCity(0) = "Istanbul"
202 toCity(1) = "Baku"
203 toCity(2) = "London"
204 priceArray(0, 0) = 888.97
205 priceArray(0, 1) = 987.98
206 priceArray(0, 2) = 299.99
207 priceArray(1, 0) = 745.75
208 priceArray(1, 1) = 923.45
209 priceArray(1, 2) = 235.97
210 priceArray(2, 0) = 967.87
211 priceArray(2, 1) = 1128.98
212 priceArray(2, 2) = 496.98
213 priceArray(3, 0) = 750.75
214 priceArray(3, 1) = 950.75
215 priceArray(3, 2) = 250.98
216 End Sub
217 Private Sub btnClear_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnClear.Click
```

The screenshot shows the Visual Studio IDE with the following components:

- Code Editor:** Contains VB.NET code for `frmAirFare.vb`. The code includes:
 - Initialization of `priceArray` with values: `priceArray(1, 2) = 235.97`, `priceArray(2, 0) = 967.87`, `priceArray(2, 1) = 1120.98`, `priceArray(2, 2) = 496.98`, `priceArray(3, 0) = 750.75`, `priceArray(3, 1) = 950.75`, `priceArray(3, 2) = 250.98`.
 - `btnClear_Click` event handler: `txtFrom.Clear()`, `txtTo.Clear()`, `txtFare.Clear()`.
 - `btnExit_Click` event handler: `Me.Close()`.
 - `btnWorthlessTotal_Click` event handler: A `For Each` loop over `priceArray` with a `Next` statement, followed by `txtWorthlessTotal.Text = FormatCurrency(totFare)`.
 - `btnAnother_Click` event handler: A `For` loop over `arrayCol` (0 to 2) and `arrayRow` (0 to 3), with a `Next` statement, followed by `txtWorthlessTotal.Text = FormatCurrency(wkTotal)`.
- Annotations:**
 - Blue text: "Using the four each to go through the items in the area. The current line is eachFare." points to the `For Each` loop in `btnWorthlessTotal_Click`.
 - Blue text: "Doing the same thing with nested for statements." points to the `For` loop in `btnAnother_Click`.
- Toolbox:** Shows a message: "There are no usable controls in this group. Drag an item onto this text to add it to the toolbox."
- Solution Explorer:** Shows the project structure: `totalairfare` (Solution), `My Project`, `AssemblyInfo.vb`, and `frmAirFare.vb`.
- Properties Window:** Shows the `Text` property of a selected control.
- Taskbar:** Shows the Windows taskbar with the Start button, search bar, and various application icons. The system tray shows the time as 1:01 PM on 3/5/2020.

```
1 Public Class Form1
2
3
4 Dim products = {{"television", 500.0, 225.0},
5                 {"computer", 1200.0, 475.0},
6                 {"laptop", 899.0, 325.0},
7                 {"tablet", 699.0, 300.0}}
8
9
10 Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
11     ListBox1.Items.Add("Description: " & products(0)(0) & " price $" & products(0)(1) & " cost $" & products(0)(2))
12     ListBox1.Items.Add("Description: " & products(1)(0) & " price $" & products(1)(1) & " cost $" & products(1)(2))
13     ListBox1.Items.Add("Description: " & products(2)(0) & " price $" & products(2)(1) & " cost $" & products(2)(2))
14     ListBox1.Items.Add("Description: " & products(3)(0) & " price $" & products(3)(1) & " cost $" & products(3)(2))
15 End Sub
16
17 Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles Button2.Click
18     For i = 0 To 3
19         ListBox2.Items.Add("Description: " & products(i)(0) & " price $" & products(i)(1) & " cost $" & products(i)(2))
20     Next
21 End Sub
22 End Class
```

In class exercise - two problems to solve
This is the answer to the first problem.
If you did not try it in class, please try it at home.

Form1

List Box 1

List Box 2

Display List Box 1

Display List Box 2

The screenshot displays the Visual Studio IDE with a VB.NET code file named 'Form1.vb'. The code defines a class 'Form1' with a 2D array of product data and two event handlers for buttons. The first handler iterates through the array and adds formatted strings to 'ListBox1'. The second handler uses a 'For' loop to add formatted strings to 'ListBox2'. Below the code, a preview of the application 'Form1' is shown, featuring two list boxes and two buttons labeled 'Display List Box 1' and 'Display List Box 2'. The list boxes contain the output of the code, showing product descriptions, prices, and costs.

```
1 Public Class Form1
2
3
4     Dim products = {{("television", 500.0, 225.0)},
5                     {("computer", 1200.0, 475.0)},
6                     {("laptop", 899.0, 325.0)},
7                     {("tablet", 699.0, 300.0)}}
8
9
10    Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click
11        ListBox1.Items.Add("Description: " & products(0)(0) & " price $" & products(0)(1) & " cost $" & products(0)(2))
12        ListBox1.Items.Add("Description: " & products(1)(0) & " price $" & products(1)(1) & " cost $" & products(1)(2))
13        ListBox1.Items.Add("Description: " & products(2)(0) & " price $" & products(2)(1) & " cost $" & products(2)(2))
14        ListBox1.Items.Add("Description: " & products(3)(0) & " price $" & products(3)(1) & " cost $" & products(3)(2))
15    End Sub
16
17    Private Sub Button2_Click(sender As System.Object, e As System.EventArgs) Handles Button2.Click
18        For i = 0 To 3
19            ListBox2.Items.Add("Description: " & products(i)(0) & " price $" & products(i)(1) & " cost $" & products(i)(2))
20        Next
21    End Sub
22 End Class
```

Form1

ListBox 1	ListBox 2
Description: television price \$500 cost \$225	Description: television price \$500 cost \$225
Description: computer price \$1200 cost \$475	Description: computer price \$1200 cost \$475
Description: laptop price \$899 cost \$325	Description: laptop price \$899 cost \$325
Description: tablet price \$699 cost \$300	Description: tablet price \$699 cost \$300