<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structures</td>
<td>1</td>
</tr>
<tr>
<td>What is a Structure?</td>
<td>2</td>
</tr>
<tr>
<td>Syntax</td>
<td>2</td>
</tr>
<tr>
<td>Example 1</td>
<td>4</td>
</tr>
<tr>
<td>Array of Structures</td>
<td>5</td>
</tr>
<tr>
<td>Including an Array in a Structure</td>
<td>7</td>
</tr>
<tr>
<td>ReDim</td>
<td>8</td>
</tr>
<tr>
<td>Example 2</td>
<td>9</td>
</tr>
<tr>
<td>OUTPUT</td>
<td>10</td>
</tr>
<tr>
<td>Example 3</td>
<td>11</td>
</tr>
</tbody>
</table>
What is a Structure?

! A **structure** is heterogeneous aggregate of data elements.

! Visual Basic allows programmers to define their own complex data types using the so called ‘Structure’ statement.

! Structures allow the programmer to combines a variety of different fields into a single unit.

! By default a Structure is declared as Public.

! Structures are somewhat similar to a CLASSES, and somewhat different than ARRAYS.

! Syntax:

```vbnet
[Public|Private] Structure   StructureName
    Dim FirstField As Datatype
    Dim SecondField As Datatype
    . . .
End Structure
```
Example:

' Create a new Data type using the ‘Structure’

    Public Structure Employee
        Dim name As String
        Dim HourlyWages As Double
        Dim HoursWorked As Double
        Dim GrossPay As Double
    End Structure

' Declaring a variable based on the Structure

    Dim employee1 As Employee

' Fields are referenced using the dot notation

    employee1.name = "John Smith"
    employee1.HourlyWages = 8.75
    employee1.HoursWorked = 38
Example 1:

Module Module1

Public Structure Employee
    Dim name As String
    Dim HourlyWages As Double
    Dim HoursWorked As Double
    Dim GrossPay As Double
End Structure

Sub Main()
    Dim employee1 As Employee

    employee1.name = "John Smith"
    employee1.HourlyWages = 8.75
    employee1.HoursWorked = 38

    Print(employee1)

    Console.ReadLine()
End Sub

Private Sub Print(ByVal TheEmployee As Employee)
    Console.WriteLine()
    Console.WriteLine("Employee Name        : {0}", TheEmployee.name)
    Console.WriteLine("Employee Hourly Wages: {0,4:c}", TheEmployee.HourlyWages)
    Console.WriteLine("Employee Hours Worked: {0}", TheEmployee.HoursWorked)
    Console.WriteLine("Employee Gross Pay   : {0,6:c}", TheEmployee.GrossPay)
End Sub

End Module
Module Module1

Public Structure Employee
  Dim name As String
  Dim HourlyWages As Double
  Dim HoursWorked As Double
  Dim GrossPay As Double
End Structure

Sub Main()
  Dim WorkGroup(4) As Employee        ' Create an array of 5 employees

  WorkGroup(0).name = "John Smith"
  WorkGroup(0).HourlyWages = 8.75
  WorkGroup(0).HoursWorked = 38
  WorkGroup(0).GrossPay = WorkGroup(0).HoursWorked * WorkGroup(0).HourlyWages

  WorkGroup(1).name = "Sue Thomas"
  WorkGroup(1).HourlyWages = 12.75
  WorkGroup(1).HoursWorked = 20

  Print(WorkGroup(0))     ' Call the Print() Procedure that accepts on employee as its parameter
  Print(WorkGroup(1))     ' Print the second employee

  Console.WriteLine()

  Console.WriteLine("Print the Entire Workgroup of employees")
  Print(WorkGroup)        ' Call the overloaded Print() procedure, which accepts the entire workgroup as its parameter

  Console.ReadLine()
End Sub

'Print()
'-------
'Accepts an Employee as a parameter
Private Sub Print(ByVal TheEmployee As Employee)
  Console.WriteLine()
  Console.WriteLine("Employee Name        : {0}", TheEmployee.name)
  Console.WriteLine("Employee Hourly Wages: {0,4:c}", TheEmployee.HourlyWages)
  Console.WriteLine("Employee Hours Worked: {0}", TheEmployee.HoursWorked)
  Console.WriteLine("Employee Gross Pay   : {0,6:c}", TheEmployee.GrossPay)
End Sub

'Print()
'-------
'Overloaded Print() procedure, which accepts the entire workgroup as its parameter
Private Sub Print(ByVal TheWorkGroup() As Employee)
  Dim EmpNo As Integer

  Console.WriteLine()
For EmpNo = 0 To TheWorkGroup.GetUpperBound(0)
    Console.WriteLine("Employee Name : {0}", TheWorkGroup(EmpNo).name)
    Console.WriteLine("Employee Hourly Wages: {0,4:c}",
        TheWorkGroup(EmpNo).HourlyWages)
    Console.WriteLine("Employee Hours Worked: {0}",
        TheWorkGroup(EmpNo).HoursWorked)
    Console.WriteLine("Employee Gross Pay : {0,6:c}",
        TheWorkGroup(EmpNo).GrossPay)
Next EmpNo

End Sub
End Module
<table>
<thead>
<tr>
<th>Employee Name</th>
<th>John Smith</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee Hourly Wages</td>
<td>$8.75</td>
</tr>
<tr>
<td>Employee Hours Worked</td>
<td>38</td>
</tr>
<tr>
<td>Employee Gross Pay</td>
<td>$332.50</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Employee Name</th>
<th>Sue Thomas</th>
</tr>
</thead>
<tbody>
<tr>
<td>Employee Hourly Wages</td>
<td>$12.75</td>
</tr>
<tr>
<td>Employee Hours Worked</td>
<td>20</td>
</tr>
<tr>
<td>Employee Gross Pay</td>
<td>$255.00</td>
</tr>
</tbody>
</table>

Print the Entire Workgroup of employees

- **Employee Name**: John Smith
  - **Employee Hourly Wages**: $8.75
  - **Employee Hours Worked**: 38
  - **Employee Gross Pay**: $332.50

- **Employee Name**: Sue Thomas
  - **Employee Hourly Wages**: $12.75
  - **Employee Hours Worked**: 20
  - **Employee Gross Pay**: $255.00

- **Employee Name**:
  - **Employee Hourly Wages**: $0.00
  - **Employee Hours Worked**: 0
  - **Employee Gross Pay**: $0.00

- **Employee Name**:
  - **Employee Hourly Wages**: $0.00
  - **Employee Hours Worked**: 0
  - **Employee Gross Pay**: $0.00

- **Employee Name**:
  - **Employee Hourly Wages**: $0.00
  - **Employee Hours Worked**: 0
  - **Employee Gross Pay**: $0.00
Including an Array in a Structure

! Arrays can be included as elements within a Structure declaration.

! VB does not, however, allow you to declare the size of the array within the Structure declaration.

! The programmer must use the ReDim statement to dynamically define the size of the array at runtime.
**ReDim**

The *ReDim* statement is used to dynamically re-declare the size of an array.

**Syntax:**

```
ReDim [Preserve] ArrayName(UpperSubscript)
```

The keyword “Preserve” can be used to preserve original data in the array. (More on this later)

**Example:**

```vbnet
Public Structure Employee
    Dim name As String
    Dim HourlyWages As Double
    Dim HoursWorked As Double
    Dim GrossPay As Double

    Dim Dependents() As String  'Dependents() is an array of strings
End Structure
```

```vbnet
Dim employee1 As Employee
ReDim employee1.Dependents(2)  'Re-dimension the dependents array
employee1.name = "Tom"
employee1.Dependents(0) = "Tom Jr."
employee1.Dependents(1) = "Jill"
employee1.Dependents(2) = "Sue"
```
Example 2:

Module Module1

Public Structure Employee
    Dim name As String
    Dim HourlyWages As Double
    Dim HoursWorked As Double
    Dim GrossPay As Double
    Dim Dependents() As String
End Structure

Sub Main()
    Dim employee1 As Employee
    employee1.name = "Tom Smith"
    employee1.HourlyWages = 8.75
    employee1.HoursWorked = 38

    ReDim employee1.Dependents(2) 'Re-dimension the dependents array.

    employee1.Dependents(0) = "Tom Jr."
    employee1.Dependents(1) = "Jill"
    employee1.Dependents(2) = "Sue"
    Print(employee1)

    '------------------------------
    Console.WriteLine()
    Console.WriteLine("ReDim array to size 4 and Preserve the existing data")
    ReDim Preserve employee1.Dependents(3) 'Re-dimension, and Preserve the existing data.

    employee1.Dependents(3) = "Jack"
    Print(employee1)

    '------------------------------
    Console.WriteLine()
    Console.WriteLine("ReDim to size 5 without Preserving existing data")
    ReDim employee1.Dependents(4) 'Redimension, without Preserveing the existing data.

    employee1.Dependents(4) = "Ann"
    Print(employee1)
    Console.ReadLine()
End Sub

Private Sub Print(ByVal TheEmployee As Employee)
    Dim Index As Integer
    Console.WriteLine()
    Console.WriteLine("Employee Name : {0}", TheEmployee.name)
    Console.WriteLine("Employee Hourly Wages: {0,4:c}", TheEmployee.HourlyWages)
    Console.WriteLine("Employee Hours Worked: {0}", TheEmployee.HoursWorked)
    Console.WriteLine("Employee Gross Pay : {0,6:c}", TheEmployee.GrossPay)
For Index = 0 To TheEmployee.Dependents.GetUpperBound(0)
    Console.WriteLine("Dependent Name : {0}", TheEmployee.Dependents(Index))
Next
End Sub
End Module
Employee Name        : John Smith
Employee Hourly Wages: $8.75
Employee Hours Worked: 38
Employee Gross Pay   : $332.50
    Dependent Name : Tom Jr.
    Dependent Name : Jill
    Dependent Name : Sue

**ReDim array to size 4 and Preserve the existing data**
Employee Name        : John Smith
Employee Hourly Wages: $8.75
Employee Hours Worked: 38
Employee Gross Pay   : $332.50
    Dependent Name : Tom Jr.
    Dependent Name : Jill
    Dependent Name : Sue
    Dependent Name : Jack

**ReDim to size 5 without Preserving existing data**
Employee Name        : John Smith
Employee Hourly Wages: $8.75
Employee Hours Worked: 38
Employee Gross Pay   : $332.50
    Dependent Name : 
    Dependent Name : 
    Dependent Name : 
    Dependent Name : 
    Dependent Name : Ann
Example 3:

Creating two new user defined structures: (Employee and MarriedEmployee)

ReDim the array within a structure.

Module Module1

Public Structure Employee
    Dim name As String
    Dim HourlyWages As Double
    Dim HoursWorked As Double
    Dim GrossPay As Double
End Structure

Public Structure MarriedEmployee
    Dim Self As Employee 'A married employee is first an employee
    Dim SpouseName As String
    Dim Dependents() As String 'A married employee may have 0 or more dependents
End Structure

Sub Main()

    Dim employee1 As Employee
    employee1.name = "John Smith"
    employee1.HourlyWages = 8.75
    employee1.HoursWorked = 38
    Print(employee1)

    Dim employee2 As MarriedEmployee
    employee2.Self.name = "Mary Johnson"
    employee2.SpouseName = "Tom"

    ReDim employee2.Dependents(2)  'Redimension the dependents array.

    employee2.Dependents(0) = "Tom Jr."
    employee2.Dependents(1) = "Jill"
    employee2.Dependents(2) = "Sue"
    Print(employee2)
'-----------------------------
Console.WriteLine()
Console.WriteLine("ReDim array to size 4 and Preserve the existing data")
    ReDim Preserve employee2.Dependents(3)  'Re-dimension, and Preserve the existing data.
employee2.Dependents(3) = "Jack"
Print(employee2)

'-----------------------------
Console.WriteLine()
Console.WriteLine("ReDim to size 5 without Preserving existing data")
    ReDim employee2.Dependents(4)     'Re-dimension, without Preserving the existing data.
employee2.Dependents(4) = "Ann"
Print(employee2)  
Console.ReadLine()
End Sub 

'Overloaded Print Procedure for printing 'Employee' information.
Private Sub Print(ByVal TheEmployee As Employee)
    Console.WriteLine()
    Console.WriteLine("Employee Name        : {0}", TheEmployee.name)
    Console.WriteLine("Employee Hourly Wages: {0,4:c}", TheEmployee.HourlyWages)
    Console.WriteLine("Employee Hours Worked: {0}", TheEmployee.HoursWorked)
    Console.WriteLine("Employee Gross Pay   : {0,6:c}", TheEmployee.GrossPay)
End Sub

'Overloaded Print Procedure for printing 'MarriedEmployee' information.
Private Sub print(ByVal TheEmployee As MarriedEmployee)
    Dim Index As Integer
    Print(TheEmployee.Self)
    Console.WriteLine(" Employee Spouse    : {0}", TheEmployee.SpouseName)
    For Index = 0 To TheEmployee.Dependents.GetUpperBound(0)
        Console.WriteLine(" Dependent Name : {0}", TheEmployee.Dependents(Index))
    Next
End Sub
End Module