Problem Solving Using Computers

Structures in VB.NET

Department of Computer and Information Sciences
Adapted from Drs. Surma, Hakimzadeh, Adaikkalavan, and Zhang
What do we need to learn in order to write computer programs?

- Fundamental programming constructs:
  - Variables,
  - Arithmetic operators,
  - Input and output
  - Conditionals,
  - Loops,
  - Arrays,
  - Subroutines and functions,
  - Structures, classes and objects,
  - Files
Why do we need Structures?

- A structure provides the means to group together logically related data elements.

- A structure is heterogeneous aggregate of data items.

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

Syntax:

[Public|Private] Structure StructureName

Dim Field1 As DataType
Dim Field2 As DataType

...

End Structure

Accessibility

The name of the structure

Elements of the structure
To define a new data type called Employee:

```
Public Structure Employee
    Dim Name As String
    Dim HourlyWage As Double
    Dim HoursWorked As Double
    Dim GrossPay As Double
End Structure
```
Declaring a variable using the new Employee structure

Dim employee1 As Employee

A new variable of type Employee is created.
Accessing the new structure

- Fields are referenced using the **dot** notation

```
employee1.Name = "Tom Hanks"
employee1.HourlyWage = 16.8
employee1.HoursWorked = 48
```

We store values into the elements of the structure
How do we use Structure?

- Employee Structure
Example 1

Module Module1
    Public Structure Employee
        Dim Name As String
        Dim HourlyWage As Double
        Dim HoursWorked As Double
        Dim GrossPay As Double
    End Structure

Sub Main()
    Dim employee1 As Employee

    employee1.Name = “Tom Hanks”
    employee1.HourlyWage = 12.6
    employee1.HoursWorked = 36

End Sub

End Module
Example 2

Module Module1
    Public Structure Employee
        Dim Name As String
        Dim HourlyWage As Double
        Dim HoursWorked As Double
        Dim GrossPay As Double
    End Structure

    Sub Main()
        Dim employee2, employee3, employee4 As Employee

        employee2.Name = "Susan Thomas"
        employee3.Name = "Tom Hanks"

    End Sub

End Module
Creating an Array of Structures

Let’s create an array of 5 employees

Dim workGroup(4) As Employee

Why 4, not 5 here?
Storing employee info in the array of employees

Sub Main()
    Dim workGroup(4) As Employee

    workGroup(0).Name = “Tom Hanks”
    workGroup(0).HourlyWage = 12.6
    workGroup(0).HoursWorked = 36
    workGroup(0).GrossPay = employee1.HourlyWage * employee1.HoursWorked

    workGroup(1).Name = “Susan Thomas”
    workGroup(1).HourlyWage = 10.0
    workGroup(1).HoursWorked = 20
    workGroup(1).GrossPay = employee1.HourlyWage * employee1.HoursWorked

    workGroup(2).Name = “Steve Hills”
    workGroup(2).HourlyWage = 16.0
    workGroup(2).HoursWorked = 60
    workGroup(2).GrossPay = employee1.HourlyWage * employee1.HoursWorked

End Sub
What are some common operations that one may perform on an employee structure?

1. Read information into the structure (e.g., insert/update)
2. Print the employee information
3. Calculate the Gross Pay for the employee
4. Retrieve information from the structure
   - GetName()
   - GetHourlyWage()
   - GetHoursWorked()
   - GetGrossPay()
Private Sub ReademployeeInfo(ByRef theEmployee As Employee)

    Console.Write("Please enter Employee's Name : ")
    theEmployee.Name = Console.ReadLine()

    Console.Write("Please enter Employee's Hourly Wage : ")
    theEmployee.HourlyWage = CDbl(Console.ReadLine())

    Console.Write("Please enter Employee's Hours Worked : ")
    theEmployee.HoursWorked = CDbl(Console.ReadLine())

    Console.WriteLine(vbCrLf)

End Sub
Private Sub PrintEmployeeInfo(ByVal theEmployee As Employee)

    Console.WriteLine("Employee Name :{0}", theEmployee.Name)
    Console.WriteLine("Employee Hourly Wage :{0:c}", theEmployee.HourlyWage)
    Console.WriteLine("Employee Hours Worked:{0}", theEmployee.HoursWorked)
    Console.WriteLine("Employee Gross Pay   :{0:c}" , theEmployee.GrossPay)

End Sub

*note {0:c}  will display number as a currency with leading $
Private Sub CalculateGrossPay(ByRef theEmployee As Employee)

    theEmployee.GrossPay = theEmployee.HourlyWage * theEmployee.HoursWorked

End Sub
Private Function GetName(ByVal theEmployee As Employee) As String

    Return (theEmployee.Name)

End Function
Private Function GetHourlyWage(ByVal theEmployee As Employee) As Double

Return (theEmployee.HourlyWage)

End Function
Private Function GetHoursWorked(ByVal theEmployee As Employee) As Double

    Return (theEmployee.HoursWorked)

End Function
Private Function GetGrossPay(ByVal theEmployee As Employee) As Double

    Return (theEmployee.GrossPay)

End Function
Putting it all together…

- We have a complete program using structures
  - EmployeeStructure
Main Program:

Module Module1
    Public Structure Employee
        Dim Name As String
        Dim HourlyWage As Double
        Dim HoursWorked As Double
        Dim GrossPay As Double
    End Structure

Sub Main()
    Dim employee1 As Employee

    ReademployeeInfo(employee1)
    PrintEmployeeInfo(employee1)

    CalculateGrossPay(employee1)
    PrintEmployeeInfo(employee1)

    Console.WriteLine(GetName(employee1))
    Console.WriteLine(GetHourlyWage(employee1))
    Console.WriteLine(GetHoursWorked(employee1))
    Console.WriteLine(GetGrossPay(employee1))

End Sub

' The subroutines and functions are inserted here…