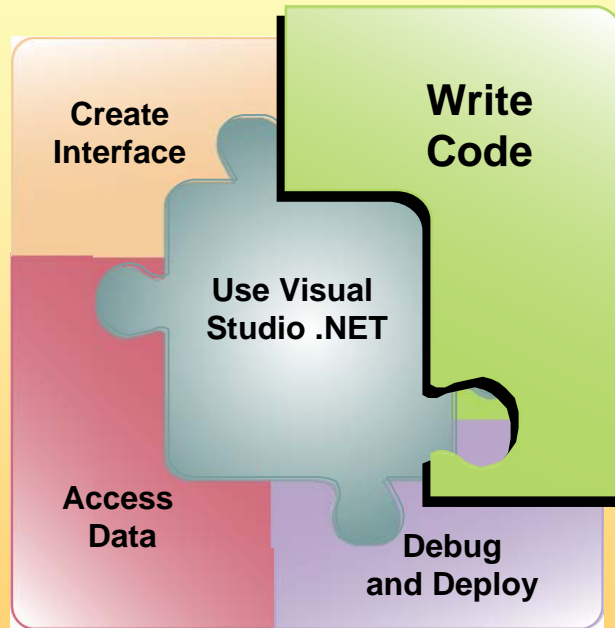


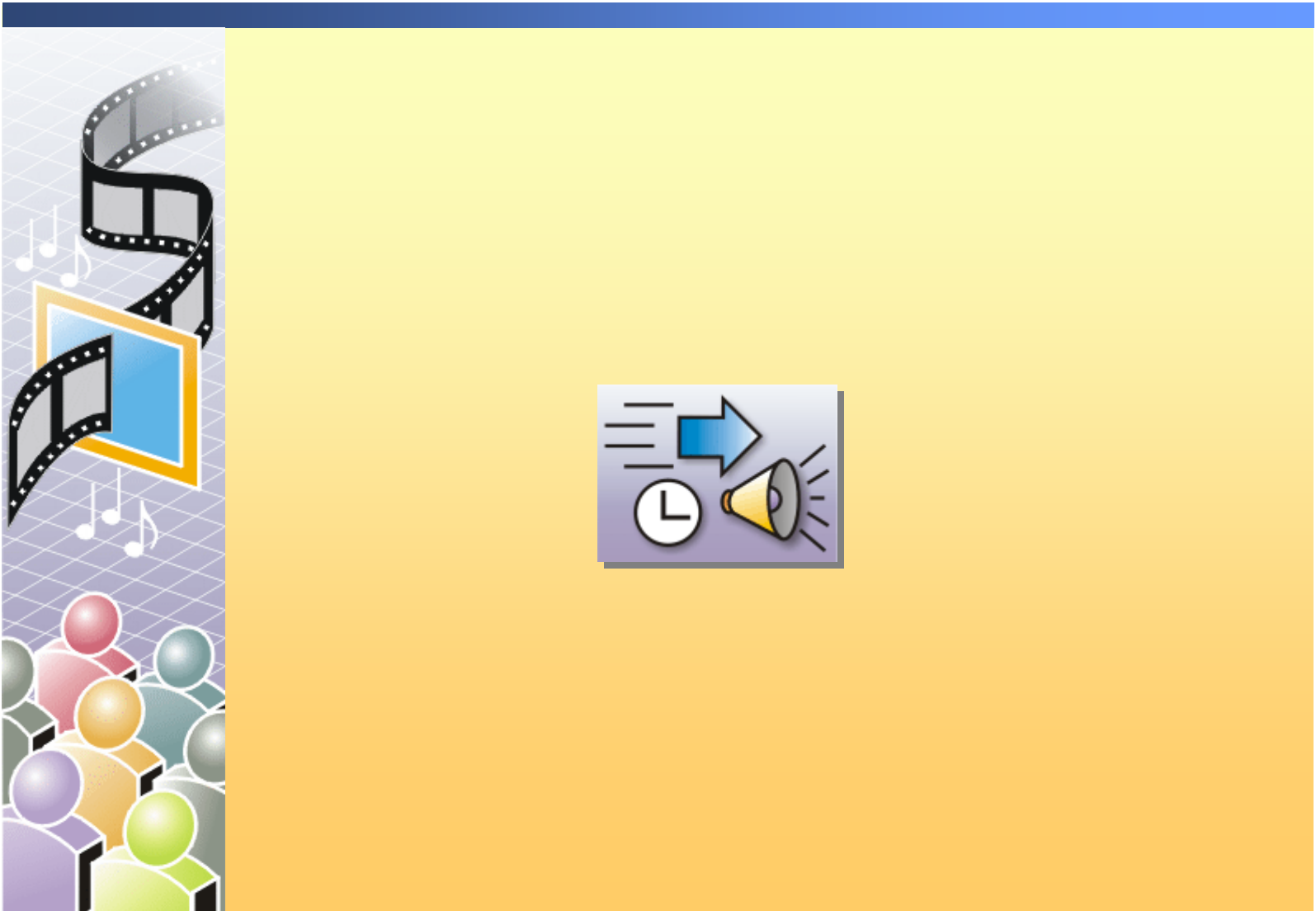
# **Module 3: Using Variables and Arrays**

# Overview



- Introduction to Data Types
- Using Variables
- Variable Scope
- Converting Data Types
- Creating and Using Structures
- Storing Data in Arrays

# Multimedia: What Are Variables?

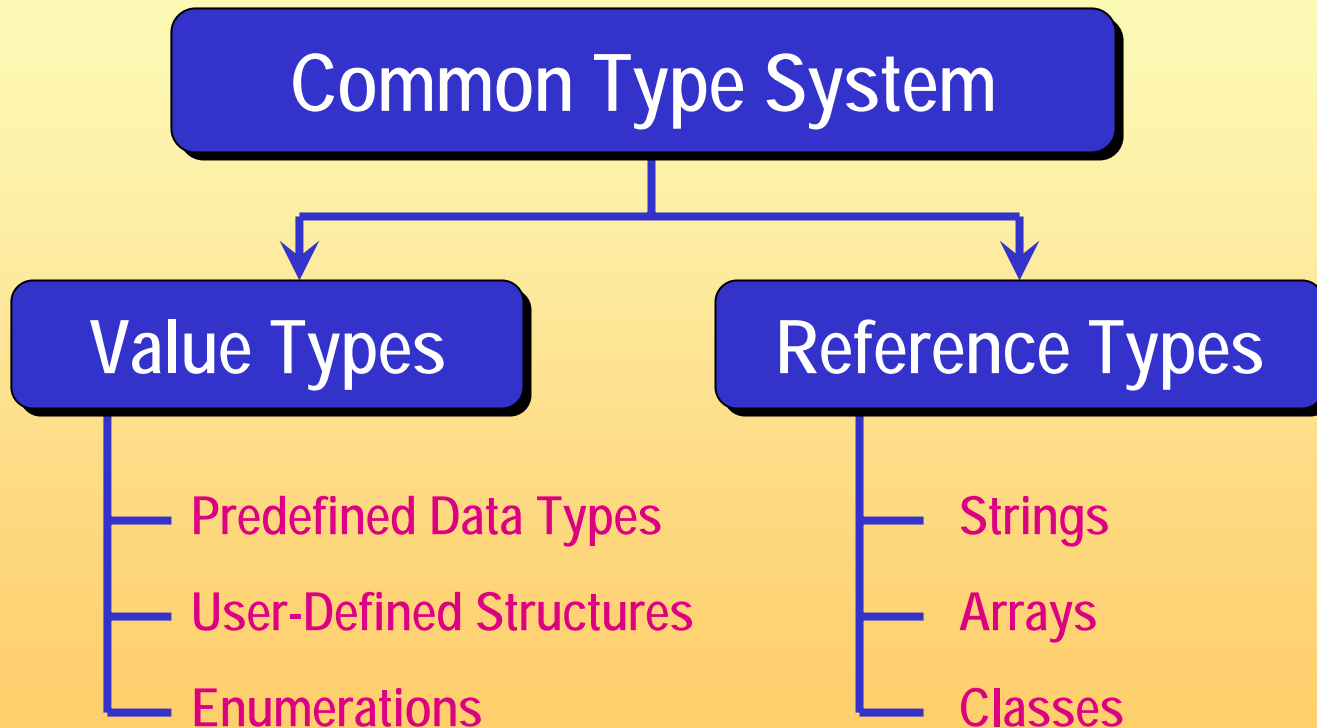


# Lesson: Introduction to Data Types

- Common type system
- Value types
- Reference types

# What Is the Common Type System?

Defines how types work in the common language runtime



# Data Types

Visual Basic .NET type	Storage size	Range of values
Boolean	2 bytes	True or False
Date	8 bytes	0:00:00 on January 1, 0001 to 11:59:59 P.M. on December 31, 9999
Decimal	16 bytes	Up to 29 significant digits, with values up to $7.9228 \times 10^{28}$ (signed)
Double	8 bytes	-4.94065645841246544E-324 to +1.79769313486231570E+308 (signed)
Integer	4 bytes	-2,147,483,648 to +2,147,483,647 (signed)
Single	4 bytes	-3.4028235E+38 to 1.401298E-45 (signed)
String	Varies	0 to approximately 2 billion Unicode characters

# How to Choose a Data Type

Select data type...	To handle...	CTS type	Example
Boolean	True or false conditions	Value	True
Short, Integer, Long, Byte	Whole integers	Value	23 (Integer)
Single, Double, Decimal	Numbers with integers and fractional parts	Value	9456.72 (Decimal)
Date	Date and time values	Value	02/12/2003 12:30:42 A.M.
String	Printable and displayable characters	Reference	"House"
Object	A pointer to the value of an object	Reference	myClass myPerson

# Practice: Choosing Data Types



**1** Analyze the data examples

**2** Consider the size and type of data

**3** Select the most compact data type

# Lesson: Using Variables

## Tasks

- 1** Name the variable
- 2** Declare the variable
- 3** Assign a value to the variable
- 4** Use the variable

# What Are Variables?

- Variables store values that can change when an application is running
- Variables have six basic elements:

Element	Description
Name	The word that identifies the variable in code
Address	The memory location at which the value is stored
Data type	The type and starting size of data the variable can store
Value	The value at the variable's address
Scope	The set of all code that can access and use the variable
Lifetime	The period of time for which a variable is valid

# How to Name Variables

## ■ Naming rules

- Start with an alphabetic character or underscore
- Do not use spaces or symbols
- Do not use keywords such as **Integer**

## ■ Examples of variable names

- `CustomerName` (PascalCasing)
- `accountBalance` (camelCasing)

# How to Declare Variables

- Syntax to declare variables
  - `Dim variableName As Type`
- Examples of value-type variables

```
Dim numberBooks As Integer  
Dim squareFootage As Single
```

- Examples of reference-type variables

```
Dim myForm As Form  
Dim userInput As String
```

# How Option Explicit Affects Variables

## ■ Option Explicit On (default)

- Forces you to explicitly declare variables before using them
- Reduces logic errors and makes code easier to maintain
- Results in faster code execution

## ■ Option Explicit Off

- Allows you to implicitly use variables without declaring them
- Increases likelihood of naming conflicts and unexpected behavior caused by spelling errors
- Results in slower code execution

# How to Assign Values to Variables

- You can:
- Assign a value to a variable after you declare it

```
Dim birthday As Date  
birthday = #3/9/1974#
```

- Assign a value to a variable when you declare it

```
Dim birthday As Date = #3/9/1974#
```

# How to Use Variables

You can use variables to:

- Store values from expressions
- Store user input
- Store objects
- Store property values
- Return values
- Display output

# Variables vs. Constants

Variables	Constants
Declare with <b>Dim</b>	Declare with <b>Const</b>
Values <b>change</b> while application is running	Values <b>stay the same</b> while application is running
Use <b>more</b> memory than constants	Use <b>less</b> memory than variables

Syntax to declare a constant:

```
Const constantName As Type
```

# Practice: Find the Bugs



**1** Dim 12Count As Integer

**2** Dim Number For Double

**3** Const Son's Birthday As Day

**4** Dim Error.Message As Text

**5** Dim \$CurrentExpenses With Decimal

# Lesson: Variable Scope

Public Module or Class

**Public *A* As Integer**

Variable *A* can be accessible to other projects within the solution

Friend Module or Class

**Friend *B* As Date**

Variable *B* is accessible anywhere within the project

Private Module or Class

**Private *c* As String**

Variable *c* is accessible anywhere within the module

Procedure or Block

**Dim *d* As Integer**

Variable *d* is accessible only within the procedure or block

# What Is Scope?

**Definition:** Scope is the set of all code that can refer to a variable by its name

## Factors that affect scope

Where you declare the variable

Block

Procedure

Module,  
Class, or  
Structure

The access level of the variable container

Private

Public

The access level of the variable

Friend

# How to Declare Local Variables

Where to declare	Keyword	Access modifier	Scope
In block	Dim	None	Block level
In procedure	Dim	None	Procedure level

## Example of local variable: block level

```
If x < > 0 Then
    Dim blockNumber As Integer
    blockNumber = x + 1
End If
```

## Example of local variable: procedure level

```
Sub ShowMessage_Click( )
    Dim myVariable As String
    ' Insert code to add functionality
End Sub
```

# How to Declare Static Variables

- Where: Declare inside a block or procedure
- Syntax: Use Static keyword (no access modifier)
  - *Static variableName As Type*
- Example

```
Sub AddItem_Click( )  
    Static items As Integer  
    ' Add 1 to the counter  
    items += 1  
    MessageBox.Show ("The count is now " & items)  
End Sub
```

# How to Declare Module Variables

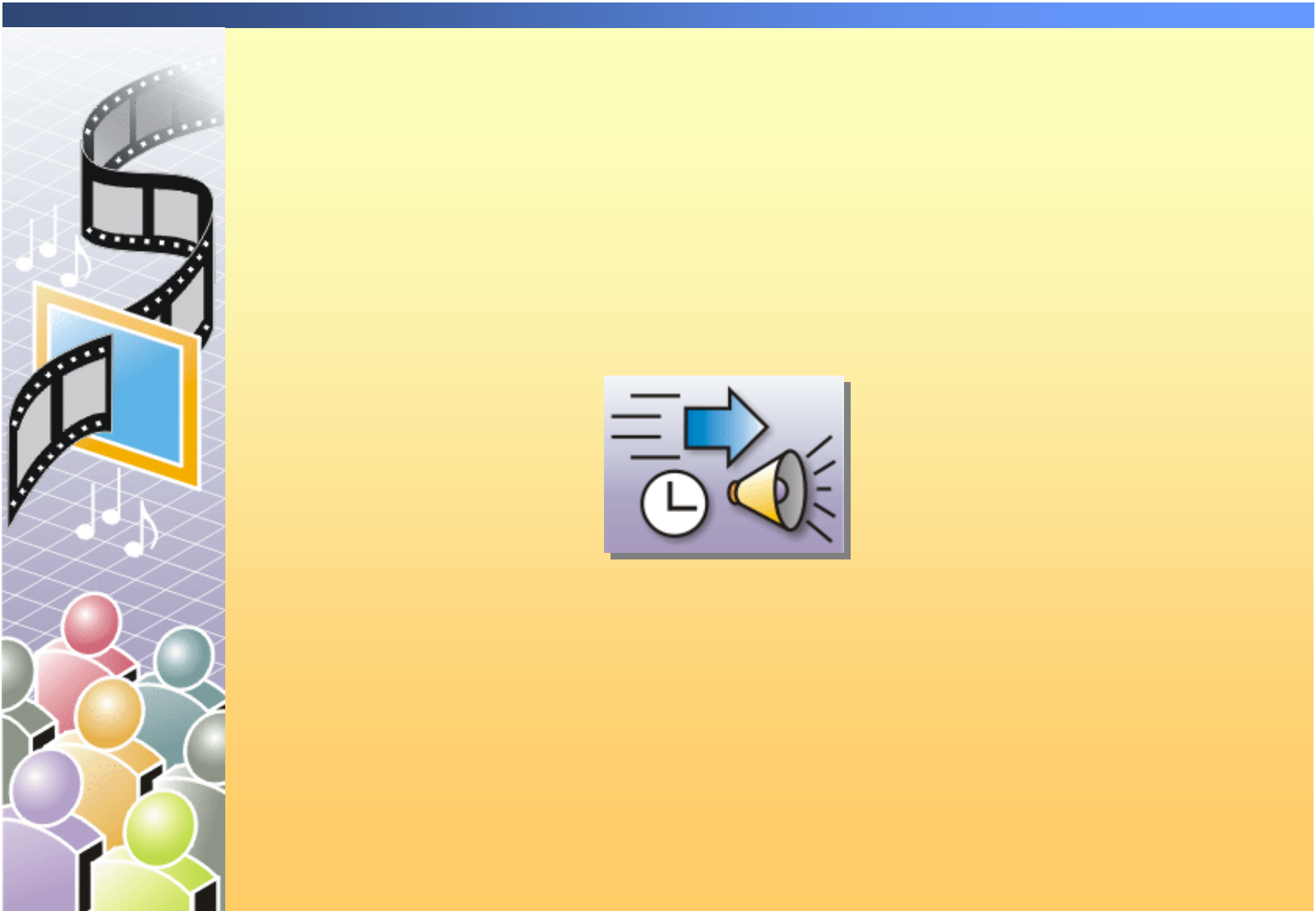
- Declare in module, class, or structure

Use access modifier ...	Scope
Private	Module
Friend	Project
Public	Solution

- Examples

```
Private myModuleMessage As String  
Friend myProjectMessage As String  
Public mySolutionMessage As String
```

# Multimedia: How to Set Access Levels for Variables



# Practice: Setting Access Levels for Variables



- 1** Examine the starter code to find an undeclared variable
- 2** Declare the variable in several different locations to achieve different levels of scope
- 3** Determine what access modifier, if any, to use when you declare the variable

# Lesson: Converting Data Types

- What Are Conversion Functions?
- How to Explicitly Convert Data Types
- How Implicit Data Conversion Works

# What Are Conversion Functions?

**Definition:** Conversion functions allow you to convert values from one data type to another data type

Integer Value  
1234

becomes

CStr

String Value  
"1234"

Double Value  
567.9894

CInt

Integer Value  
568

String Value  
"February 12, 1992"

CDate

Date Value  
#2/12/92#

# How to Explicitly Convert Data Types

Syntax: *VariableName* = *CFunction*(*Expression*)

## Example

- 1 Declare a variable as String data type  
`Dim myString As String`
- 2 Declare another variable as Integer data type  
`Dim myInteger As Integer`
- 3 Assign a value to the string variable  
`myString = "1234"`
- 4 Convert the string value to an integer value  
`myInteger = CInt(myString)`

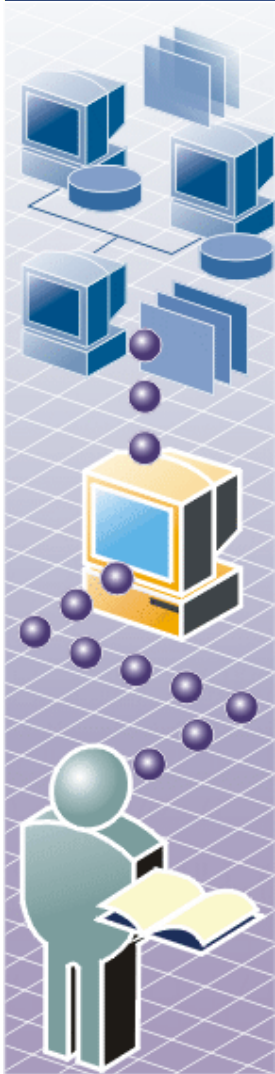
# How Implicit Data Conversion Works

- Data types are automatically converted
- No special syntax is required in code
- Example of implicit data conversion:

```
Dim sequence As String
Dim number As Integer
' ...
sequence = "1234"
number = sequence
' The value in sequence is implicitly converted
  to an Integer
```

- Disadvantages of implicit data conversion:
  - Can yield unexpected results
  - Code runs slower
- Option Strict disallows any implicit narrowing type conversions

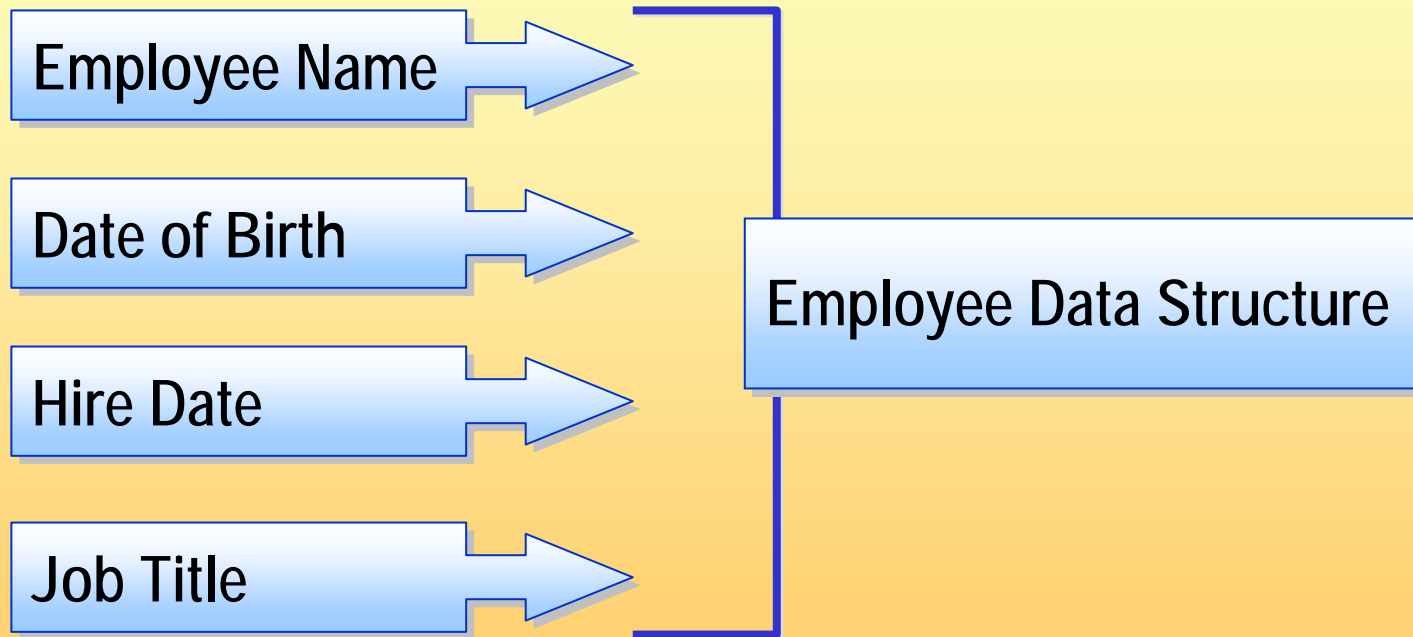
# Lab 3.1: Creating and Using Variables



- Exercise 1: Using Static Variables
- Exercise 2: Creating, Using, and Converting Variables

# Lesson: Creating and Using Structures

Group related information → into a single structure



# What Are Structures?

- Composite data types
- Used to create user-defined value types
- Members can be variables, properties, methods, or events
- Example of a user-defined structure:

```
Public Structure Employee  
    Public FirstName As String  
    Public LastName As String  
    Public HireDate As Date  
    Public JobTitle As String  
    Private Salary As Decimal  
End Structure
```

- Examples of predefined structures: *Point, Size, Color*

# How to Declare Structures

- Inside a module, file, or class (not in a procedure)
- Syntax to declare structures:

```
AccessModifier Structure StructureName  
    ' Declare structure members here  
End Structure
```

- Where access modifier is:
  - **Public** for unrestricted access
  - **Protected** for access only within its own class
  - **Friend** for access anywhere in the application or assembly
  - **Private** for access only within its declaration context
- Do not assign values to data members in the declaration

# How to Use Structures

## Procedure

- 1** Declare a structure
- 2** Declare a variable of that structure type
- 3** Assign values to the data members
- 4** Write code to use the structure members

# Practice: Creating and Using Structures



- 1** Declare a structure
- 2** Declare a variable as the structure type
- 3** Assign values to the structure members
- 4** Write code to use the structure members
- 5** Run and test the application

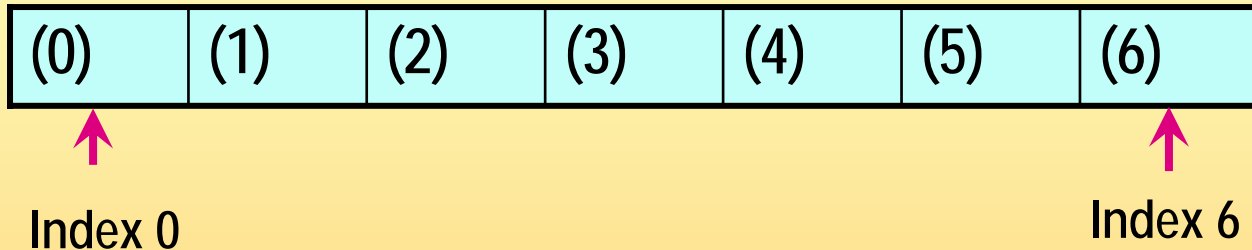
# Lesson: Storing Data in Arrays

- What Is an Array?
- How to Declare a Single-Dimension Array
- How to Use Multidimensional Arrays
- How to Resize Arrays

# What Is an Array?

- **Definition: An array is a series of data elements**

- All elements in an array have the same data type
- Individual elements are accessed by using integer indexes



- **Example**

- To declare an integer array with seven elements:

```
Dim countHouses(6) As Integer
```

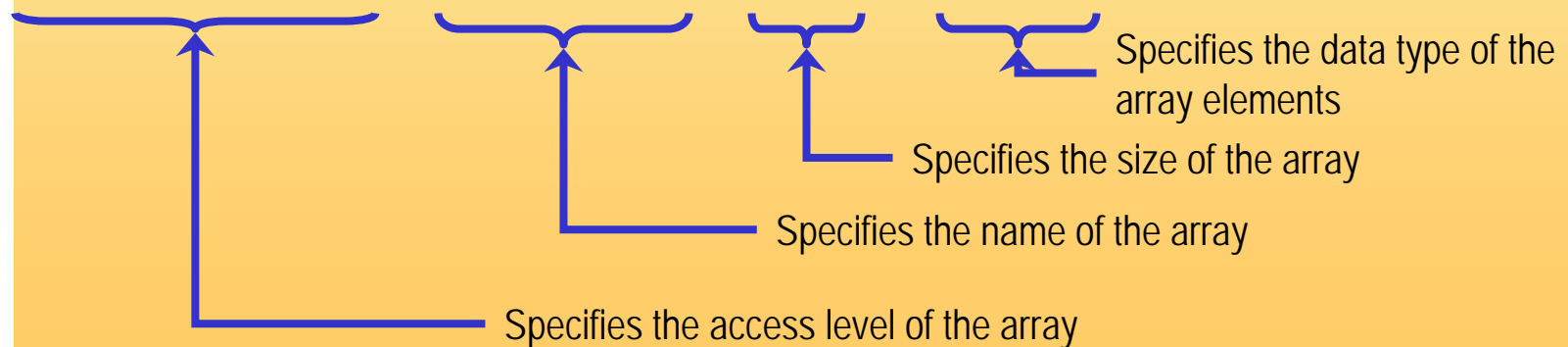
- To access the third element in the array:

```
TextBox1.Text = CStr(countHouses(2))
```

# How to Declare a Single-Dimension Array

- You declare an array by specifying the:
  - Name of the array
  - Size (number of elements)
  - Data type of the array elements
  - Access modifier (if needed)

*AccessModifier ArrayName(Size) As Type*



# How to Use Multidimensional Arrays

- Specify all dimensions and elements
- Total elements = product of all the sizes
- To declare a multidimensional array variable:
  - Add one pair of parentheses after the variable name
  - Place commas inside the parentheses to separate the dimensions
  - Begin declaration with **Dim** statement or an access modifier
- Example:

```
Public ThreeDimensions(3,9,14) As Double  
    ' Three-dimensional array
```

# How to Resize Arrays

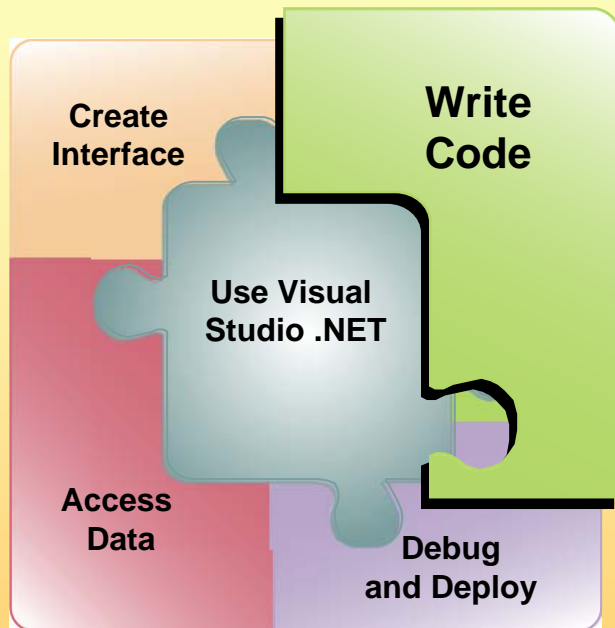
- You can resize an array at any time
- Use the ReDim statement
- Syntax to resize arrays:

```
ReDim existingArray(NewSize)
```

- Example:

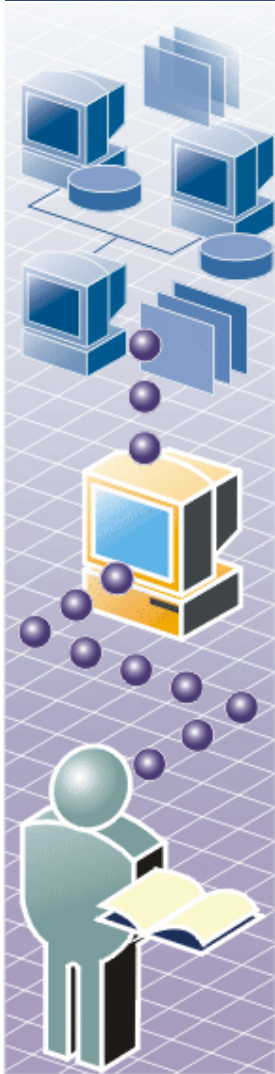
```
Dim myArray(,) ' Declare array  
ReDim myArray(3, 5) ' Redimension array
```

# Review



- Introduction to Data Types
- Using Variables
- Variable Scope
- Converting Data Types
- Creating and Using Structures
- Storing Data in Arrays

# Lab 3.2: Using Structures and Arrays



- Exercise 1: Creating a Structure
- Exercise 2: Creating and Using Arrays