# VB.Net

Introduction

# What is Programming?

- **Programming** is the process of taking an algorithm and encoding it into a notation – a programming language.
- Without an **algorithm** there can be no program
- Fundamentals of programming:-
  - Sequence
  - Selection
  - Iteration
  - Invocation (method calls)

# What is this “VB.Net”?

- VB - Visual Basic
- .Net –
  - Is a programming framework created by Microsoft.
  - A bunch of code that the programmer can call without having to write it explicitly.
  - Runs primarily on Microsoft Windows.

# What is Visual Basic.Net?

- Object-oriented computer programming language.
- Implemented on the .NET Framework.
- Everything in VB.NET is an object.
- Has access to all the libraries in the .Net Framework

# Types of Applications Using .Net

- Windows Applications
- Web Applications
- Web Services

# Integrated Development Environment (IDE)

- Software suite that consolidates the basic tools developers need to write and test software
- Visual Basic 2010 Express
- Some IDE support add-ons

# VB.Net Program Structure

- Namespace declaration
- A class or module
- One or more procedures
- Variables
- The Main procedure
- Statements & Expressions
- Comments

# Object Naming Conventions

| Object | Prefix | Example |
|---|---|---|
| Form | frm | *frmHello* |
| Button | btn | *btnExit* |
| Label | lbl | *lblMessage* |
| Text | txt | *txtName* |
| Menu | mnu | *mnuSave* |
| Check box | chk | *chkChoice* |
| Option/Radio Button | opt | *optSelect* |

# Hands on Programming

Message Formatter

# Data Types

| VB Data Type | Use For | Storage Size in bytes |
|---|---|---|
| Boolean | True or False values | 2 |
| Byte | 0 to 255, binary data | 1 |
| Char | Single Unicode character | 2 |
| Date | 1/1/0001 through 12/31/9999 | 8 |
| Decimal | Decimal fractions, such as dollars and cents | 16 |
| Single | Single-precision floating-point numbers with six digits of accuracy | 4 |
| Double | Double-precision floating-point numbers with 14 digits of accuracy | 8 |
| Short | Small integer in the range −32,768 to 32,767 | 2 |
| Integer | Whole numbers in the range −2,147,483,648 to +2,147,483,647 | 4 |
| Long | Larger whole numbers | 8 |
| String | Alphanumeric data: letters, digits, and other characters | Varies |
| Object | Any type of data | 4 |

# Choose Data Types Wisely…

- Prevents memory issues
- Prevents runtime errors
- Avoid unnecessary conversions

# Variable Name Convention

- No spaces
- No Reserved/Keywords
- Unique name for variable in same scope
- Descriptive
- Importance:
  - To make programming easy and maintainable.

# Examples of Good Variable Names

Dim strStudentFName As String

Const dblDISCOUNT_RATE As Double= 0.15

# Scope of Variables

- Block-Level variables usually used with repetition and decision structures.
- Local variable maybe used only within the procedure.
- Class-level variable can be accessed from all procedures.

# Classroom Exercise

Programming Example For Scope of Variables

# Statements and Expressions

- Assignment statement – (=)
  - intSum = intSum + intNewVal
  - Left side is always assigned the value of the right side
  - Follows the below order:-

    Exponentiation (^)
    Unary identity and negation (+, –)
    Multiplication and floating-point division (*, /)
    Integer division (\)
    Modulus arithmetic (Mod)
    Addition and subtraction (+, –)
    String concatenation (&)
    Arithmetic bit shift (<<, >>)

# Statements and Expressions

- Continuation – (_)
  - To be put on end of incomplete sentence
  - Important because VB.Net does not have code separating characters.

**Example**:

```
Months = Log(Final * IntRate / Deposit + 1) _
/ Log(1 + IntRate)
```

# Statements and Expressions

- Commenting – (Rem) or (‘)
  - To be used for leaving comments
  - Code commenting is important for reusability and for maintainability.
  - Example:

    *Rem This variable will be used to store the total marks of the student*

    *‘This variable will be used to store the total marks of the student*

# Operators

- ^ : Exponentiation
- * : Multiplication
- / : Division
- \ : Integer division
- Mod : Modulus
- + : Addition
- - : Subtraction

# Comparison

- `>` : Greater than
- `<` : Less than
- `>=` : Greater than or equal to
- `<=` : Less than or equal to
- `=` : Equal to
- `<>` : Not equal to
  - Results will be **True** or **False**

# Logical Operators

- Not - Logical not
- And - Logical and
- Or - Logical or

# Boolean Operators

| X | Y | XY (X And Y) | X + Y (X Or Y) |
|---|---|---|---|
| False | True | False | True |
| False | False | False | False |
| True | True | True | True |
| True | False | False | True |

# Decision Structure

**There are 3 main types of decision structures:-**

- **If…Then…Else Construction**
- **Select…Case Construction**
- **Try…Catch…Finally Construction**
  - **We will focus on If…Then….Else Construction**

# If….Then..Else (Example)

**Example:**

If *the sun is shining* Then (condition)
    *go to the beach* (action to take if condition is true)
Else
    *go to class* (action to take if condition is false)
End If

# Select…Case (Example)

```
Dim number As Integer = 5

Select Case number
    Case 1 To 5
        strReturn = "Between 1 and 5, inclusive"
    Case 6, 7, 8
        strReturn = "Between 6 and 8, inclusive"
    Case 9 To 10
        strReturn = "Equal to 9 to 10"
    Case Else
        strReturn = "Not between 1 and 10, inclusive"
End Select
```

# Try…Catch…Finally (Example)

```
Dim x As Integer = 5
Dim y As Integer = 0
' Set up structured error handling.
Try
    ' Cause a "Divide by Zero" exception.
    x = x \ y

    MessageBox.Show("end of Try block")
Catch ex As Exception
    MessageBox.Show(ex.Message)
Finally
    MessageBox.Show("in Finally block")
End Try
```

# Classroom Exercise

Programming Example For Decision Structures

# Loop Structures

- While Loops
- Do Loops
- For Loops
- For Each Loops
  - We will be focusing on Do and For Loops.

# While Loops

- Checks the conditions first before the initial iteration.

```
While someCondition
    '…Statements to execute
End While
```

# Do Loop

- Similar to while loop, but it can also have:-
  - Until option – statements to execute until its true
  - Check the condition after the first iteration.

```
Do {While | Until} Condition
        ' Statements in loop.
Loop

Or

Do
        ' Statements in loop.
Loop {While | Until} Condition
```

# For Loop

- Performs the loop a set number of times

```
For index As Integer = 1 To 5
    Debug.Write(index.ToString & " ")
Next
Debug.WriteLine("")
' Output: 1 2 3 4 5
```

# For Each Loops

- Repeats a group of statements for each element in a collection.

```
' Create a list of strings by using a
' collection initializer.
Dim lst As New List(Of String) _
    From {"abc", "def", "ghi"}

' Iterate through the list.
For Each item As String In lst
    Debug.Write(item & " ")
Next
Debug.WriteLine("")
'Output: abc def ghi
```

# Exit Statement

- Exit { Do | For | Function | Property | Select | Sub | Try | While }

# Classroom Exercise

Programming Example For Loops

# ByVal vs ByRef

- ByVal – passing by value: A copy of the object is passed in the procedure. Any changes made to the variable in the procedure does not affect the original value.
- ByRef – passing by reference: A pointer of the object is passed in the procedure. Changing the variable in the procedure will affect the original variable.

# Procedures

- Sub Procedure
  - Does not return a value
- Function Procedure
  - Returns a value

# Sub Procedure Example

```
Sub tellOperator(ByVal task As String)
    Dim stamp As Date
    stamp = TimeOfDay()
    MsgBox("Starting " & task & " at " & CStr(stamp))
End Sub
```

```
tellOperator("file update")
```

# Function Procedure Example

```
Private Function getIntSqrd(ByVal val As Integer) As Integer
    Return val * val
End Function
```

```
Dim value As Integer = 5
txtOutput.Text = getIntSqrd(value)
```

# Classroom Exercise

Programming Example For Procedures…ByVal and ByRef

# Fixing Errors

- Syntax Error
- Run-time Error
- Logic Error

# Classroom Exercise

Programming Example For Fixing Errors

# Classroom Discussion

- Difficulties you faced?
  - Your solution
- Difficulties your students face?
  - Your solution to their problem
- Any new innovations used?

# Thank You!!

The End...