CSE 1530

Introduction to Computer Use II: Programming

Winter 2006 (Section M)

Topic C: Control Structures - Selection

Wednesday, February 3 2006

Bill Kapralos

Overview (1):

- **Before We Begin**
  - Some administrative details
  - Some questions to consider
- **Boolean Operators**
  - Introduction
  - The operators
- **Validating User Input**
  - Introduction

Before We Begin

Administrative Details (1):

- **Lab Exercise 3-3**
  - Exercise has been graded and will be distributed back to you after today's lecture
  - Has been graded /1 (e.g., either it is correct or not correct)
- **Reminder**
  - You should be working on Ex 4-4 this week
  - Test 1 will be held February 8 2006
    - More details today!

Some Questions to Consider (1):

- What is an If and If/Else statement ?
- What is a nested If statement ?
- With a nested If statement, can more than one Boolean expression be True ?
- How do we obtain random numbers in Visual Basic ? And are these numbers really random ?

Boolean Operators

Introduction (1):

- **So Far, Boolean Expressions → Limited Use**
  - Basically, our Boolean expressions compare values of two variables (values) and return True or False
  - But there are times we need to make comparisons between more than two variables (values) or there are times we want to combine Boolean expressions
    - Can this be done with the knowledge we have so far regarding Boolean expressions ?

```
If (age < 12) or (age >65) then
   price = 10
```

Introduction (2):

- **Allow Us to Combine Boolean Expressions**
  - Mathematical operators such as +, -, / and *, take two numerical values and produces a numerical result
  - Similar to the mathematical operators, a Boolean operator takes two Boolean values (operands) and produces a new Boolean value
    - Boolean operators perform operations on Boolean data (types) → returns a Boolean value
  - Typically, Boolean operators are binary → take two operands
    - But they can be unary (e.g., take one operand only)

Introduction (3):

- **Visual Basic Boolean Operators**
  - We will look at some of the common Boolean operators
    - AND, OR, NOT, XOR,
    - AND, OR and NOT are probably familiar to you as you make use of them in everyday conversations!
    - Depending on value of operands, operator output is "pre-defined"
  - In what follows, op1 & op2 are two Boolean operands
    - Boolean variables, expressions etc.

The Boolean Operators (1)

- **The AND Operator**
  - Everyday example → If the sun is shining AND the temperature is hot, I will go to the beach
  - Result is True only if both operands are True

OP1	OP2	RESULT
False	False	False
True	False	False
False	True	False
True	True	True

The Boolean Operators (2)

- **The OR Operator**
  - Everyday example → If the sun is shining OR the temperature is hot, I will go to the beach
  - Result is True if one or both operands are True

OP1	OP2	RESULT
False	False	False
True	False	True
False	True	True
True	True	True

The Boolean Operators (3)

- **The XOR (Exclusive Or) Operator**
  - Everyday example → If the sun is shining XOR the temperature is hot, I will go to the beach
  - Result is True if only one operand is True only but not if both are True

OP1	OP2	RESULT
False	False	False
True	False	True
False	True	True
True	True	False

The Boolean Operators (4)

- **The NOT Operator**
  - Negation operator
    - Negates the value of the operand

OP1	RESULT
False	True
True	False

The Boolean Operators (5):

- **Boolean Operators and Visual Basic**
  - Examples → try this on your own!

```
Dim var1 As Boolean
Dim var2 As Boolean
Dim var3 As Boolean

var1 = True
var2 = False

var3 = var1 AND var2
var3 = var1 OR var2
var3 = var1 XOR var2
var3 = NOT var1
```

The Boolean Operators (6):

- **Boolean Operators and Visual Basic (cont.)**
  - Examples → operands are now expressions
    - Each expression is evaluated resulting in a Boolean value and then Boolean operator performed

```
Dim var1 As Double
Dim var2 As Double
Dim var3 As Boolean

var1 = 10
var2 = 50

var3 = (var1 < var2) AND (var2 > var1)
var3 = (var1 < var2) OR (var2 > var1)
```

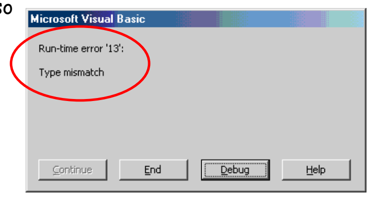
Validating User Input

Introduction (1):

- **User Input Is Not Always Valid!**
  - From the exercises you have worked on up to this point, you are probably well aware that as a programmer you cannot make assumptions regarding the validity of user entered input!
    - Although we may require input to be numeric, there is no guarantee user will not enter letters!
    - Anything can be assigned to the Text property of a Textbox → the problem occurs when we try to use the entered data!

Introduction (2):

- **User Input Is Not Always Valid! (cont.)**
  - Up until this point, invalid user entered input results in the program "crashing"!
    - Program execution abruptly ends and you are presented with a runtime error message box indicating so


Microsoft Visual Basic

Run-time error '13':

Type mismatch

Introduction (3):

- **User Input Is Not Always Valid! (cont.)**
  - Typically, there is no need to have the program exit in such an abrupt manner for what may perhaps be a simple mistake on behalf of the user
    - Why not, for example, simply re-prompt the user to enter the data when there is a mistake ?
    - This is in fact the approach we will be taking → we will check user entered data to ensure it meets our requirements
    - Only perform calculations with the user entered data if it is valid otherwise, we re-prompt user

Introduction (4):

- **Using Visual Basic's Built in Functions To verify Input Data**
  - Various functions available to us to check the contents of String data → IsNumeric function
  - IsNumeric(StringData)
    - Function that takes a String argument (StringData) and checks to see if the characters comprising the String are all numeric values
    - Returns True if the characters are all numeric and False otherwise

Introduction (5):

- **Using Visual Basic's Built in Functions To verify Input Data (cont.)**
  - How can we make use of this function in Exercise 4-4 in order to ensure the values entered by the user (e.g., the "guesses") are all valid (numeric) ?
    - Hint → You will use an If/Else statement

Additional Notes

The If Statement (1):

- **Recall The Structure of the If Construct**

```
Other statements in sub-program
If (Boolean Value) Then
   statement 1
   statement 2
   …
End If
More statements
```

The statements following the If/Then line can be any valid Visual Basic statements including another If or If/Else statement!

```
If (Boolean Value 1) Then
  If (Boolean Value 2) Then
     statements
     …
  End If
End If
```