Decision Structures in VB

CMPT 110

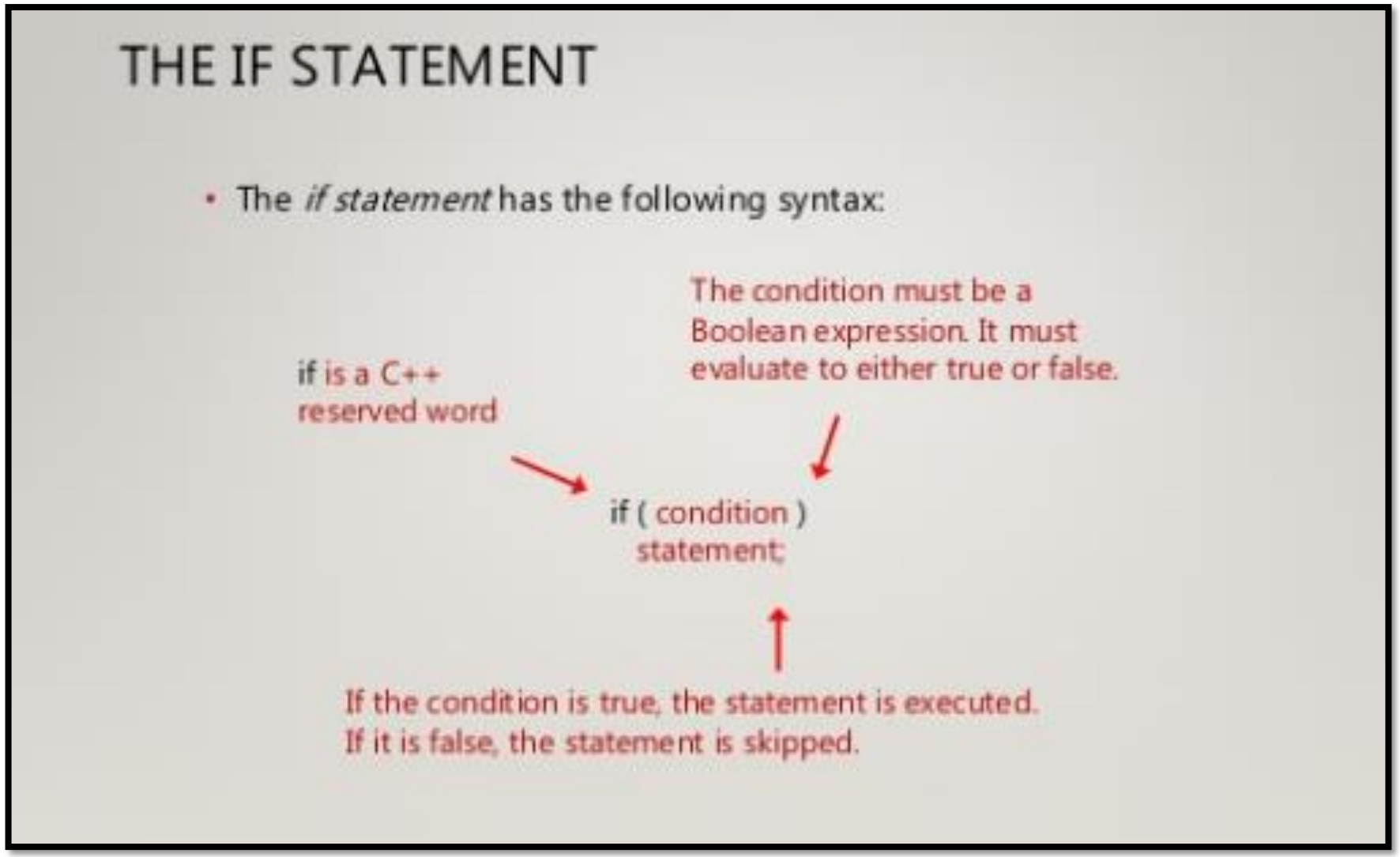
Objectives From Study Guide 7

- To learn the syntax of the two types of programming blocks provided in Visual Basic for controlling the execution sequence of a program.
- To be able to decide which of the different types of programming blocks is more appropriate in a given application.
- To be able to program effectively using either type of programming block.

Preliminary Notes on Decision Structures

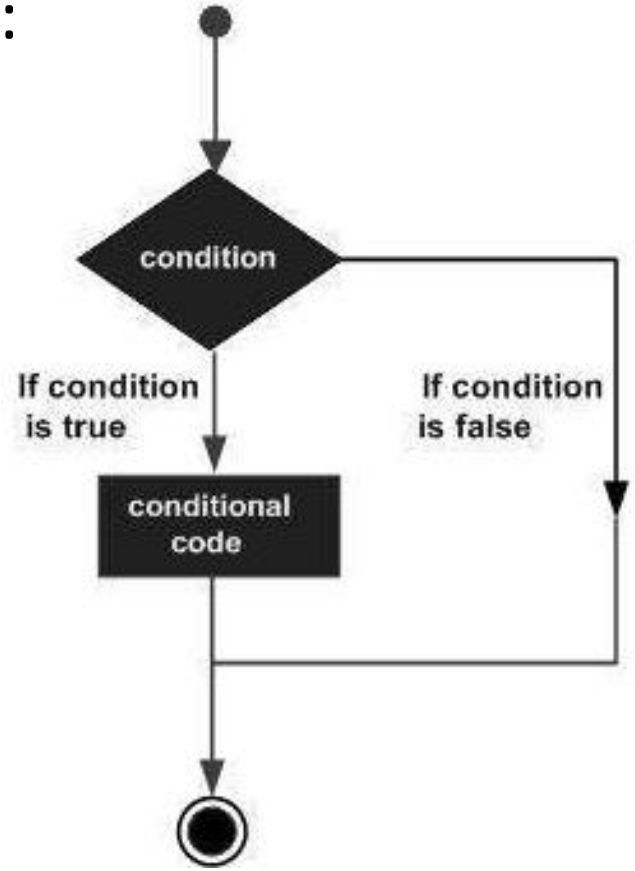
- A *decision structure* is a construct in a computer program that allows the program to make a decision resulting in a change of program state.
- The decision is made based on the Boolean outcome of a *logical test* – the result is either true or false.

Preliminary Notes on Decision Structures



Visual Basic Decision Structure

The following is the general form of a typical decision making structure found in most of the programming languages:



Visual Basic Decision Syntax

IF STATEMENT

Executes code based on a condition, the condition must evaluate true for the code to execute.

SYNTAX

```
If True Then
End If
```

EXAMPLE

```
If Year > 2010 Then
Console.WriteLine("Hello World!")
End If
```

visual-basic-Cheat-Sheet.pdf

Visual Basic Decision Syntax

If Else Statement

The If Else Statement works similar to the if statement, however if the first condition is false the else condition will execute.

Example

```
If Year < 2010 Then
    Console.WriteLine("Hello World!")
Else
    Console.WriteLine("Hello!")
End If
```


visual-basic-Cheat-Sheet.pdf

VB.Net – Types of Decisions in More Detail

Statement	Description
If ... Then statement	An **If...Then statement** consists of a boolean expression followed by one or more statements.
If...Then...Else statement	An **If...Then statement** can be followed by an optional **Else statement**, which executes when the boolean expression is false.
nested If statements	You can use one **If** or **Else if** statement inside another **If** or **Else if** statement(s).
Select Case statement	A **Select Case** statement allows a variable to be tested for equality against a list of values.
nested Select Case statements	You can use one **select case** statement inside another **select case** statement(s).

Syntax of If… Then

- **Simple 'IF' :**

Simple IF used only for one condition.

Syntax :

```
If logical_expression Then
    One or more Visual Basic statements
End If (block close)
```

Example :

```
Dim x As Integer
Dim y As Integer
x = 3
y = 4
If x > y Then
    MsgBox ("a is greater then b")
Else
    MsgBox ("b is greater then a")
End If
```


https://www.slideshare.net/blpragya/vb-decision-making-statements

If… Then… Else

Whereas If executes code based on the condition's true condition, the Else statement executes code based on the condition's false condition

Syntax :

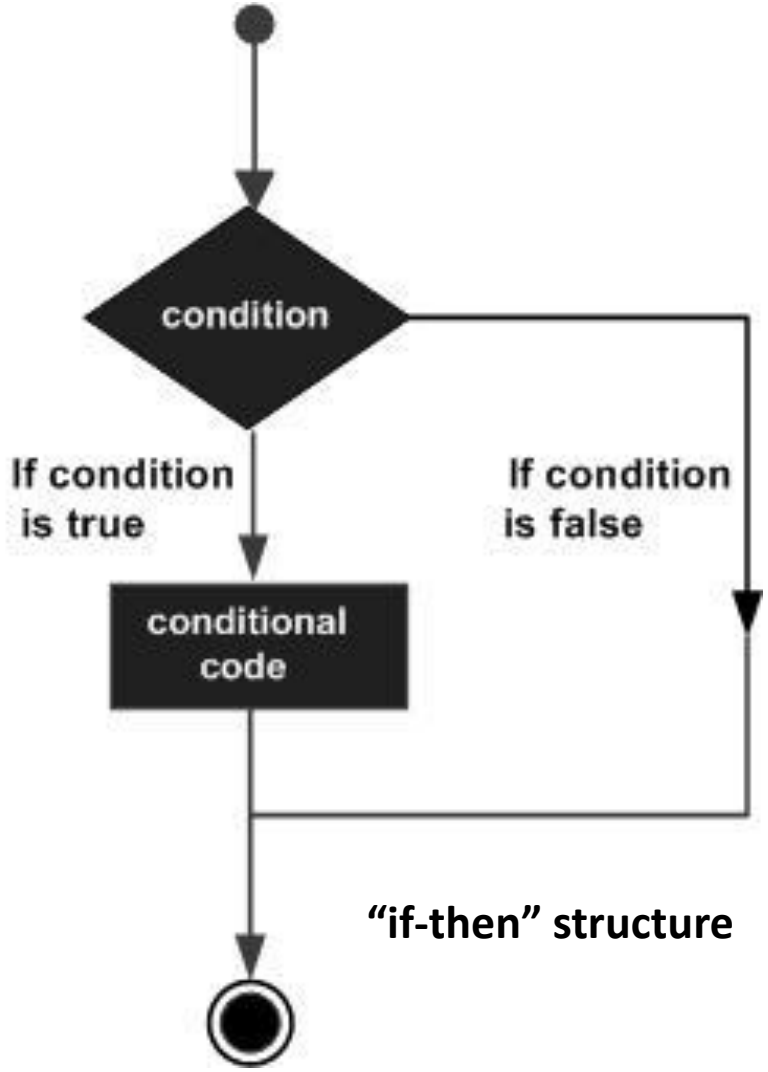
```
If logical_expression Then
    One or more Visual Basic statements
Else
    One or more Visual Basic statements
End If
```

Example :

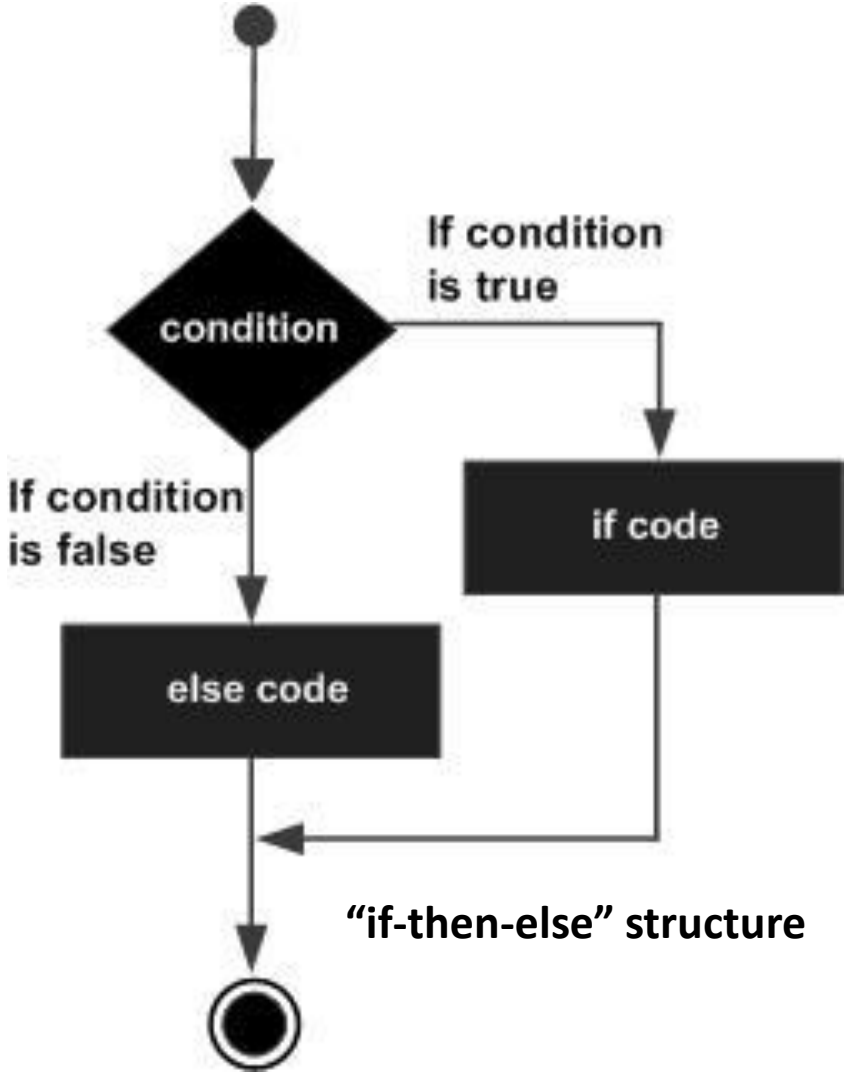
```
If (num > 0) Then
    Print "Number is Positive"
Else
    Print "Number is negative"
End If
```

https://www.slideshare.net/blpragya/vb-decision-making-statements

Distinction Between If-Then & If-Then-Else



“if-then” structure



“if-then-else” structure

Nested If Statements

Rule of Sum

28. **a)** Determine the value of the integer variable *counter* after execution of the following program segment. (Here i, j, and k are integer variables.)

```
counter := 0
for i := 1 to 12 do
   counter := counter + 1
for j := 5 to 10 do
   counter := counter + 2
for k := 15 downto 8 do
   counter := counter + 3
```

From Grimaldi

Un-nested If Statements (Sequential)

28. **a)** Determine the value of the integer variable *counter* after execution of the following program segment. (Here i, j, and k are integer variables.)

```
counter := 0
for i := 1 to 12 do
   counter := counter + 1
for j := 5 to 10 do
   counter := counter + 2
for k := 15 downto 8 do
   counter := counter + 3
```

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 (12 cycles)

5, 6, 7, 8, 9, 10 (6 cycles)

15, 14, 13, 12, 11, 10, 9, 8 (8 cycles)

From Grimaldi

Un-nested If Statements (Sequential)

28. **a)** Determine the value of the integer variable *counter* after execution of the following program segment. (Here i, j, and k are integer variables.)

```
counter := 0
for i := 1 to 12 do
   counter := counter + 1
for j := 5 to 10 do
   counter := counter + 2
for k := 15 downto 8 do
   counter := counter + 3
```

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 (12 cycles)

5, 6, 7, 8, 9, 10 (6 cycles)

15, 14, 13, 12, 11, 10, 9, 8 (8 cycles)

0 + 12(1) + 6(2) + 8(3) = 48

From Grimaldi

Nested If Statements

29. Consider the following program segment where i, j, and k are integer variables.

```
for i := 1 to 12 do
  for j := 5 to 10 do
    for k := 15 downto 8 do
      print (i - j)*k
```

a) How many times is the **print** statement executed?

b) Which counting principle is used in part (a)?

From Grimaldi

Nested If Statements

29. Consider the following program segment where i, j, and k are integer variables.

```
for i := 1 to 12 do
  for j := 5 to 10 do
    for k := 15 downto 8 do
      print (i - j)*k
```

a) How many times is the **print** statement executed?

b) Which counting principle is used in part (a)?

From Grimaldi

12x6x8=576

Nested If Statements – Provide a VB Example

Nested If Statements – Provide a VB Example

```
01  Private Sub cmdLogin_Click(ByVal sender As System.Object
02      If txtPassword.Text <> "" Then
03          If txtUser.Text = strUser Then
04              If txtPassword.Text = strPassword Then
05                  booLogin = True
06                  If booLogin = True Then
07                      MsgBox("Succesfully logged in!", Msg
08                      txtPassword.Text = ""
09                      txtUser.Text = ""
10                      booLogin = False
11                  End If
12              Else
13                  MsgBox("Password is incorrect, please tr
14                  txtPassword.Text = ""
15              End If
16          Else
17              MsgBox("Username is incorrect, plaese try ag
18
19          End If
20      Else
21          MsgBox("Please enter a password!", MsgBoxStyle.I
22      End If
23  End Sub
```

Other VB Decision Structures

From Study Guide – Unit 7

Visual Basic provides two types of programming blocks for the conditional execution of instructions:

- **IF** Block: Based on the value of a Boolean expression, an IF block permits the programmer to indicate what to do when the expression is True and when it is False.
- **Case** Block: *This block permits the programmer to perform different tasks for each value or range of values that an expression may have.*

Other VB Decision Structures

SELECT CASE

The Select Case statement is similar to a switch statement found in many other programming languages. A few points:

- Select Case evaluate one variable
- You can use some operators
- Select Case Statements are must easier to maintain then using nested if else

SYNTAX

```
Select Case variableName
    Case 1
    Case 2
    Case Else
End Select
```

visual-basic-Cheat-Sheet.pdf

Other VB Decision Structures

SELECT CASE

EXAMPLE

```
Select Case Year
      Case 2012
          Console.WriteLine("It's 2012!")
      Case 2013
          Console.WriteLine("The current year!")
      Case Year > DateTime.Now.Year
          Console.WriteLine("Year is greater than 2013")
      Case Else
          Console.WriteLine("....")
  End Select
```


visual-basic-Cheat-Sheet.pdf

Other VB Decision Structures

Try...Catch...Finally Construction

`Try...Catch...Finally` constructions let you run a set of statements under an environment that retains control if any one of your statements causes an exception. You can take different actions for different exceptions. You can optionally specify a block of code that runs before you exit the whole `Try...Catch...Finally` construction, regardless of what occurs. For more information, see Try...Catch...Finally Statement.

Decisions Based Upon Keystrokes

Decisions Based on Keystrokes – **From Unit 7**

A common decision in event-driven programming is to perform a task based on the key that is pressed on the keyboard. Pressing a key is an example of an "event" and so there is an event method associated with it. The introduction of IF blocks provides an opportunity to introduce a new method—the "KeyPress" method—extending those previously described in Unit 3.

Since the pressing of a key is often associated with typing into a TextBox, a KeyPress method can be defined for TextBoxes. For example, suppose we want to write a program that permits the user to signal the completion of the entry of data into a TextBox. With the methods discussed previously, we could use a Button for this purpose. However, an alternate way is for the user to press the "Enter" (or "Return") key on the keyboard after typing into the TextBox. To implement the signalling of the completion of data entry in this way, a suitable KeyPress event method needs to be defined.

Decisions Based on Keystrokes – **From Unit 7**

The KeyPress method detects not just the pressing of the Enter key but also the pressing of any key on the keyboard. Therefore a suitable test is required that compares the key pressed with the Enter key. How is this test done?

The answer is that when any key on the keyboard is pressed, the ANSI character codeword (Links to an external site.)Links to an external site. for that key is generated as described in Unit 3. Therefore, if we know the ANSI code for the "Enter" key (13_{10}), we can test for that codeword every time a KeyPress event occurs

See https://www.youtube.com/watch?v=ujchErnUJJA for details of implementation.

Decisions Based on Keystrokes – **From Unit 7**

The following method processes KeyPress events for a TextBox named `TextBox1`:

```
Private Sub TextBox1_KeyPress(ByVal sender As Object, _
    ByVal e As System.Windows.Forms.KeyPressEventArgs) _
    Handles TextBox1.KeyPress
    If e.KeyChar = Chr(13) Then
        data = TextBox1.Text
    Else
    End If
End Sub
```

In this example, the conditional expression is `e.KeyChar = Chr(13)`. This is a relational expression, and so is either True or False. If the relational expression is True, then the assignment statement `data = TextBox1.Text` is executed: that is, the contents of `TextBox1` is stored in the class variable called `data`. On the other hand, if the expression is False, "nothing" happens since there are no statements to execute following the Else statement.

Using Strings in If-Statements

Using Strings in If-Statements

Strings can be used in the conditions for an If statement. String variables can be compared to other string variables and string variables can be compared to string literals. The same six comparison operators used for numeric comparison are also used for strings. If

```
strRightAnswer = "T"
strStudentAnswer = "F"
```

Using Strings in If-Statements

String Comparisons

String Comparisons

Comparison	Result	Reason
If "A" = "a" Then	False	"A" and "a" are different
If "B" < "C" Then	True	"B" comes before "C" in ASCII
If "SHOUT" = "shout" Then	False	upper case and lower case are different
If "Cat" > "cat" Then	False	"C" comes before "c" in ASCII
If "sun" <> "moon"	True	"sun" and "moon" are different

Presentation Terminated