CT240: Programming Algorithms

Date:

16-September 2011

Topic:

Control Structures

Control Structures

- Control Structures in Programming Languages control the order of execution of instructions in the program
- Examples of control structures:
  - Statements
  - Precedence Rules
  - Selection Statements
  - Iterative Statements
  - Recursive Statements

Evaluation Order

- What is the value of

$$2*3+4$$

- When there are no parenthesis, operations are performed in the following order:
- 1)exponentiations;
  2)multiplications and divisions;
  3)additions and subtractions.
- Parentheses (brackets) should be used when necessary to clarify the meaning of an expression.

Evaluation Order

Summary of precedence:

()	Inner to outer, left to right
^	Left to right in expression
* /	Left to right in expression
+ -	Left to right in expression

Examples …

- (3 + (2 * (5 - 1)))
- 3 + 2 * 5 - 1
- (3 + 2) * (5 – 1)
- 2 + 3 – 4
- 3 – 4 + 2
- 2 * 3 / 4
- 3 / 4 * 2

Examples …

1.1.

```
a = 5
b = -2
a = a + b
b = b – a
```

1.2.

```
a = 31
b = 16
temp = a
a = b
b = temp
```

1.3.

```
a= 0
b = 7
a = a + ((b mod 2) * -3)
b = b + 4 * a
```

1.4.

```
a = -8
b = 3
temp = a + b
b = a
temp = temp + a + b
```

2. Suppose a and b have been declared as integer variables with values 3 and 5 respectively. What are the values of the following expressions:

- (a mod b) + (a mod b)
- (b – a) * (a + b)
- b – (a * a) + b
- b – (a *(a + b))

Relational Operators

- A **condition** is an expression that is either True or False.
- **Relational operators** form conditions and can be used with numbers and strings. e.g.

```
2 < 5
2 + 7 > 9 * 3
"c" < "d"
"e" >= "m"
```

Relational Operators in VB

Operator	Numeric Meaning	String Meaning
=	equal to	identical to
<>	not equal to	different from
<	less than	precedes alphabetically
>	greater than	follows alphabetically
<=	less than or equal to	precedes alphabetically or is identical to
>=	greater than or equal to	follows alphabetically or is identical to

Expressions in Conditions

- Conditions can involve variables, operators and functions.
- To determine whether a condition is true or false, first compute the numeric or string values and then evaluate the resulting assertion.
- Examples:

  Suppose a = 4, b = 3, c = "hello"

  Evaluate:

  7 = 7

  7 = a

  a > b

  “A” <= “B”

  “B” > “A”

  4 < (3+2)

  (a + b) < 2 * a

  (c.length() - b) = (a/2)

Logical Operators … AND OR NOT

Logical operators can be used to combine conditions, thus forming more complex conditions.

Let *cond1* and *cond2* be conditions, then

- *cond1* `AND` *cond2*

  is true if both *cond1* and *cond2* are true, and false otherwise.
- *cond1* `OR` *cond2*

  is true if either (or both) *cond1* or *cond2* is true, and false otherwise.
- `NOT` *cond1*

  is true if *cond1* is false, and false if *cond1* is true.

Examples:

`(2 < n) AND (n < 5)`

`(answ = "Y") OR (answ = "y")`

AndAlso and OrElse

- Two new logical operators in VB.Net
- Allow quicker evaluation of logical expressions and/or ... stopping when the outcome is ‘guaranteed’

  Examples:

```
Dim n as Integer
n = 10
(20 < n) AndAlso (n < 500)
(2 < n) OrElse (n < 500)
```

Evaluating Logical Expressions

- Where parentheses are missing, VB uses the following operator hierarchy:
  - First, all arithmetic operations are performed
  - Next, all relational operations (<,>,=) are evaluated to either true or false
  - Finally, the logical operators are applied in the order NOT, AND, OR.
  - In the event of a tie, the leftmost operator is applied first.

```
a < b + c OR d < e AND NOT f = g
```

Notes

- Conditions are evaluated to either true or false, depending on the current values of the variables contained in the conditions. These two values are called the possible **truth values** of the condition.
- A condition such as (2 < n < 5) should not be used, because VB will not evaluate it as intended. Instead, you should write (2 < n) AND (n < 5).

Selection …. motivations ….

- Write a program which calculates the entry charge to a cinema based on age, where the following prices apply for customers :

  older than 65 -> 5.50

  20 or younger -> 7.00

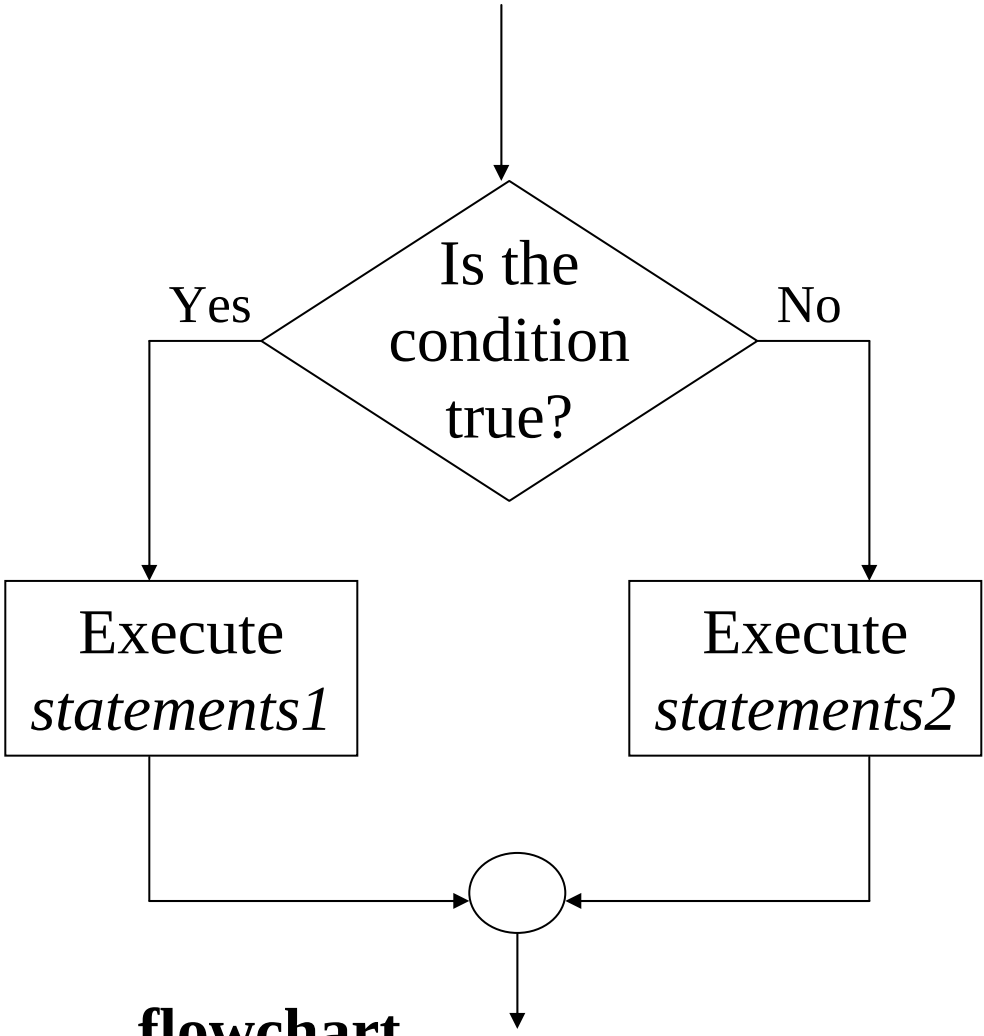
  anyone else -> 10.00

- Selection statements allow you to **specify alternatives** in programs
- An if statement can have one or more alternatives

If Blocks

- An **If Block** allows a program to select a course of action based on whether a condition is true or false.
- An If Block has the form:

```
If condition then
        statements1
    Else
        statements2
End If
```



flowchart

If-ElseIf Blocks

- An extension of the If-Block allows for more than two possible alternatives.
- An If-ElseIf Block has the form:

```
If condition1 Then
        statements1
    ElseIf condition2 Then
        statements2
    ElseIf condition3 Then
        statements3
    ...
    Else statements
End If
```

If-ElseIf Blocks

- The `If-ElseIf` block checks each condition until the first true condition is found; it then carries out the statements associated with that condition, and then skips all other conditions and goes to the `End If`.
- If none of the conditions is true, then the `Else's` statements are executed.
- In general, an If block can have
  - exactly one If-clause
  - many ElseIf-clauses
  - at most one Else clause.
- Example:
- Given two integer numbers which are entered in two text boxes: `txtNum1` and `txtNum2`, print out the largest of the numbers to a text box `txtAns`

Solution …

```
Dim num1, num2 As Integer

num1 = Val(txtNum1.text)
num2 = Val(txtNum2.text)
If num1 > num2 Then
   txtAns.Text = num1
Else If num2 > num1 Then
   txtAns.Text = num2
Else
  txtAns.Text = “The numbers are equal”
End If
```

Class Problem

- The user is asked to input a real number into the text box `txtNumber` for which the square root is to be taken. Write code so that if the user enters a negative number the message “Number can’t be negative” is displayed and the text box is cleared. Otherwise, output the square root of the number
- - use the VB.Net function `Math.Sqrt()`

Improve this if statement

```
If a < b Then
   If c < 5 Then
      txtAns.text =
       “hello”
  End If
End If
```

What is the output?

```
Dim a, b, c As
  Integer
a = 2
b = 3
c = 5
If a * b < c Then
  b = 7
  Else
  b = c * a
End If
txtAns.text = b
```

Find the errors …

```
Dim num As Decimal
num = 0.5
If (1 < num < 3) Then
      txtAns.text = “Number is between 1 and 3”
End If
```

```
Dim num As Integer
num = 6
If (num > 5 AND < 9) Then
   txtAns.Text = “yes”
Else
   txtAns.Text = “no”
End If
```

```
Dim j, k As Integer
j = 2
k = 3
If (j OR k = 4) Then
   txtAns.text =  “OK!”
End If
```

Points to Remember when Using if Statements

- used for selection
- using *elseif* construct may be easier than lots of *if*s but sometimes you cannot avoid lots of *if* statements - it depends on the problem
- be careful with the ‘else’ part of an *if*/*elseif* statement. If possible try to avoid using else
- be careful not to leave out some alternative
- can nest if statements