Al-Mustaqbal University College
Department of Medical Instrumentation Techniques Engineering
Class: Second Class
Subject: Computer Applications
Lecturer: Assistant Lecturer Miami Abdul Aziz
Lecture: Seventh Lecture

Arithmetic Expressions

Most applications require the computer to perform at least one calculation. You instruct the computer to perform a calculation using an arithmetic expression, which is an expression that contains **one** or more arithmetic operators. Figure 1-8 lists the most commonly used arithmetic operators available in Visual Basic, along with their precedence numbers. It also includes several examples of using the operators. The precedence numbers indicate the order in which the computer performs the operation in an expression. Operations with a precedence number of 1 are performed before operations with a precedence number of 2, and so on. However, you can use parentheses to override the order of precedence because operations within parentheses are always performed before operations outside parentheses.

Operator	Operation	Precedence
^	exponentiation (raises a number to a power)	1
–	negation (reverses the sign of a number)	2
*, /	multiplication and division	3
\	integer division	4
Mod	modulus (remainder) arithmetic	5
+, –	addition and subtraction	6

Note: You can use parentheses to override the normal order of precedence. For instance, in the 6 / (2 + 4) example, the parentheses indicate that the addition should be performed before the division.

Examples	Results
7 ^ 2	49
6 / 2 + 4	7
6 / (2 + 4)	1
33 \ 5	6
33 Mod 5	3

Figure 1-8 Most commonly used arithmetic operators

Although the negation and subtraction operators use the same symbol (a hyphen), there is a difference between them. The negation operator is a unary operator, which

means it requires only one operand. The expression –10 uses the negation operator to turn its one operand (the positive number 10) into a negative number. The subtraction operator, on the other hand, is a binary operator; this means it requires two operands. The expression 8 – 2 uses the subtraction operator to subtract its second operand (the number 2) from its first operand (the number 8). The integer division operator (\) divides two integers and then returns the result as an integer. Using this operator, the expression 9 \ 4 results in 2, which is the integer result of dividing 9 by 4. (If you use the standard division operator [/] to divide 9 by 4, the result is 2.25 rather than 2.) The modulus operator (sometimes referred to as the remainder operator) also divides two numbers, but the numbers do not have to be integers. After dividing the numbers, the modulus operator returns the remainder of the division. For example, the expression 9 Mod 4 equals 1, which is the remainder of 9 divided by 4. A common use for the modulus operator is to determine whether a number is even or odd. If you divide the number by 2 and the remainder is 0, the number is even; if the remainder is 1, the number is odd. Some of the arithmetic operators, such as the addition and subtraction ones, have the same precedence number. When an expression contains more than one operator having the same priority, those operators are evaluated from left to right. In the expression

45 - 20 / 5 + 15

the division is performed first, then the subtraction, and then the addition. The result of the expression is the number 56, as shown in Example 1 in Figure 1-9. You can use parentheses to change the order in which the operators in the expression are evaluated, as shown in Example 2. The parentheses tell the computer to perform the

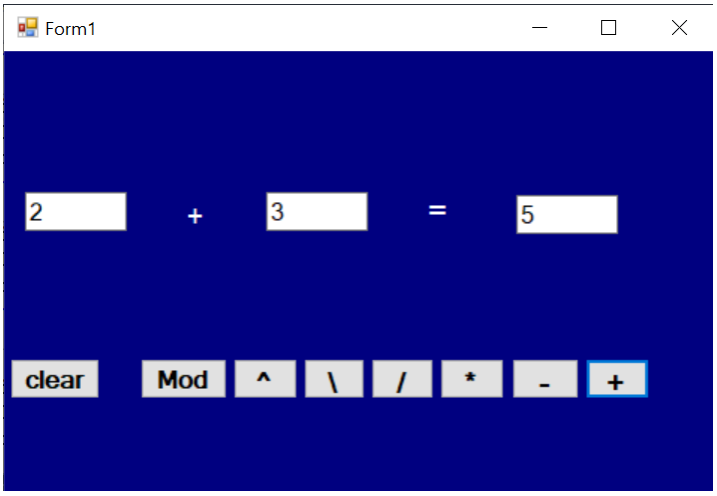
addition first, then the division, and then the subtraction; the result is 44 rather than 56.

Example 1		Example 2	
Expression: 45 – 20 / 5 + 15		Expression: 45 – 20 / (5 + 15)	
Division first	45 – 20 / 5 + 15	Addition first	45 – 20 / (5 + 15)
Subtraction next	45 – 4 + 15	Division next	45 – 20 / 20
Addition last	41 + 15	Subtraction last	45 – 1
Answer:	56	Answer:	44

Figure 1-9 Expressions containing more than one operator having the same precedence

The arithmetic expressions you enter in your code should not contain commas or special characters, such as the dollar sign or percent sign. To include a percentage in an arithmetic expression, you must use its decimal equivalent. For example, to multiply 30 by 5%, you would use the expression 30 * 0.05 (rather than 30 * 5%). Arithmetic expressions can also contain numeric variables, What is the meaning of variables? This is what we will explain in the next lecture.

Practical part:



Al-Mustaqbal University College
Department of Medical Instrumentation Techniques Engineering
Class: Second Class
Subject: Computer Applications
Lecturer: Assistant Lecturer Miami Abdul Aziz
Lecture: Seventh Lecture

```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        TextBox3.Text = Val(TextBox1.Text) + Val(TextBox2.Text)
        Label2.Text = Button1.Text
        Label2.Visible = True
    End Sub
    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        TextBox1.Clear()
        TextBox2.Text = ""
        TextBox3.Text = ""
        Label2.Visible = False
    End Sub
    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
        TextBox3.Text = Val(TextBox1.Text) - Val(TextBox2.Text)
        Label2.Text = Button3.Text
        Label2.Visible = True
    End Sub
    Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
        TextBox3.Text = Val(TextBox1.Text) * Val(TextBox2.Text)
        Label2.Text = Button4.Text
        Label2.Visible = True
    End Sub
    Private Sub Button5_Click(sender As Object, e As EventArgs) Handles Button5.Click
        TextBox3.Text = Val(TextBox1.Text) / Val(TextBox2.Text)
        Label2.Text = Button5.Text
        Label2.Visible = True
    End Sub
    Private Sub Button6_Click(sender As Object, e As EventArgs) Handles Button6.Click
        TextBox3.Text = Val(TextBox1.Text) \ Val(TextBox2.Text)
        Label2.Text = Button6.Text
        Label2.Visible = True
    End Sub
    Private Sub Button8_Click(sender As Object, e As EventArgs) Handles Button8.Click
        TextBox3.Text = Val(TextBox1.Text) ^ Val(TextBox2.Text)
        Label2.Text = Button8.Text
        Label2.Visible = True
    End Sub

    Private Sub Button9_Click(sender As Object, e As EventArgs) Handles Button9.Click
        TextBox3.Text = Val(TextBox1.Text) Mod Val(TextBox2.Text)
        Label2.Text = Button9.Text
        Label2.Visible = True
    End Sub
End Class
```

Email: miami.abdulaziz@uomus.edu.iq