Ch2 : Arithmetic

Probably every program that you will ever write will contain at least one calculation, and this is true of the majority of programs. It is not surprising therefore that Pascal and other high-level languages make calculations easy to perform. Arithmetic instructions simply involve defining what arithmetic operations are to be performed on numeric identifiers and constants.

The four common arithmetic operations: add; subtract; multiply; divide, use the symbols +, -, * and /, respectively. The samples of arithmetic operations provide in Table 2 1 assume that the following data declarations have been made.

```
Const     PI  = 3.14
DIM Length, Width, Perimeter          AS Integer
DIM Area, Radius, Gallons, Miles, Mpg AS Decimal
DIM a, b, c, x, y                     AS Decimal
```

Math Expression	**VB Program Code**
Area = Length X Width	Area = Length * Width
Area = πr^2	Area = PI * Radius *Radius
Perimeter = 2 X (Length + Width)	Perimeter = 2 * (Length + Width)
Mpg = Gallons ÷ Miles	Mpg = Gallons / Miles
X = 0	x = 0

Table 2.1.

All of the statements in Table 2.1 involve calculating a value using decimal or integer variables or a combination of decimals and integers.
VB rules concerning how such calculations how may be expressed are called assignment *c*ompatibility rules. They state that calculation which involves:

(i) a mixture of integers and decimals must be assigned to a decimal variable,

(ii) only integers may be assigned to either an integer variable or a decimal variable

Another point to note is that VB provides three divide operators

The '/' divide operator may be used with any values, decimal or integer, but *if both values are integers* then the result is a decimal value and must be assigned to a decimal variable.

The second divide operator, div, is only allowed to be used with integers. If the result of a division *does not* produce a whole number, the fractional part is ignored. In other words, the result is rounded down to the nearest integer.

The remainder, when one integer is divided by another, is produced by the Mod operator. Same examples should help to clarify these points and these am shown in Table 2.2

Operation	Example	Answer	Data Type of Answer
Decimal / Decimal	1.23 / 0.4 0.5 / 0.25	3.075 2.0	Decimal Decimal
Decimal / Integer	13.9 / 5	2.78	Decimal
Integer / Decimal	16 / 2.5	6.4	Decimal
Integer / Integer	21 / 4	5.25	Decimal
	20 / 4	5.0	Decimal
Integer DIV Integer	21 div 4	5	Integer
	20 div 4	5	Integer
	2 div 3	0	Integer
Integer DIV Decimal	Not permitted		
Decimal DIV Decimal	Not permitted		
Integer MOD Integer	21 mod 4	1	integer
	20 mod 4	0	
Integer MOD Decimal	Not permitted		
Decimal MOD Integer	Not permitted		
Decimal MOD Decimal	Not permitted		

VB uses the backslash
\ to represent DIV

Integer Division
The next example program uses integer division to convert a number of seconds into hours minutes and seconds.

```
1   Sub Main()
2         'This program converts given seconds into hours, minute, seconds

3         Const SECONDSPERMINUTE = 60
4         Const MINUTESPERHOUR = 60
5         'Variables
6         Dim Hours As Integer
7         Dim Minutes As Integer
8         Dim Seconds As Integer
9         Dim Duration As Integer
10        Dim Temp As Integer

11         'input
12         Console.Clear()
13         Console.Write("Enter Number of Seconds :")
14         Duration = Console.ReadLine
15         'process
16         Seconds = Duration Mod SECONDSPERMINUTE
17         Temp = Duration \ SECONDSPERMINUTE
18         Minutes = Temp Mod MINUTESPERHOUR
19         Hours = Temp \ MINUTESPERHOUR
20         'output
21         Console.WriteLine("----------------------")
22         Console.WriteLine("Hours    :{0}", Hours)
23         Console.WriteLine("Minutes  :{0}", Minutes)
24         Console.WriteLine("Seconds  :{0}", Seconds)
25         Console.WriteLine()
26         Console.WriteLine("  :{0}:{1}:{2}", Hours, Minutes, Seconds)

27         Console.ReadKey()
28    End Sub
```

Code Analysis

1	Start of subroutine Main ( )
2	Brief comment describing the program
3	Const. Set `SECONDSPERMINUTE = 60` A constant is a type of variable but unlike variables its value cannot be changed… I.e. it remains constant throughout the program.
4	Const. `MINUTESPERHOUR = 60`
5	Variables section
6	DIM (Declare in Memory) a location named Hours. It will be allowed to store Integer values
7	DIM (Declare in Memory) a location named Minutes. It will be allowed to store Integer values
8	DIM (Declare in Memory) a location named Seconds. It will be allowed to store Integer values
9	DIM (Declare in Memory) a location named Duration. It will be allowed to store Integer values
10	DIM (Declare in Memory) a location named Temp. It will be allowed to store Integer values
11	Input section
12	`Console.Clear()` this statement clears the console screen
13	Write the letters in quotes " " are written to the console screen. Prompt the user to enter the number of seconds.
14	Readline ( ) allowed data to be read from the user. The input is placed into the variable (memory location) named Duration.
15	Process section
16	**Seconds** is calculated by MOD division. ie, Duration MOD `SECONDSPERMINUTE.` The result is placed into location named **Seconds**
17	**Temp** is calculated by DIV division. ie, `Duration \ SECONDSPERMINUTE.` The result is placed into location named **Temp**
18	**Minutes** is calculated by MOD division. ie, `Temp MOD MINUTESPERHOUR.` The result is placed into location named **Minutes**
19	**Hours** is calculated by DIV division, `Hours = Temp \ MINUTESPERHOUR.` The result is placed into location named **Hours**
20	Output section
21	Writes a dotted line to the console.
22	The letters in quotes are written to the console screen… … Note the brackets used {0} The 0 (zero) indicates that the value held in **Hours** should appear here.
23	As above to show Minutes
24	As above to show Seconds
25	Writes a blank line to the console screen
26	This line demonstrates formatting output using several variables on one line. Note there are three items of data printed. The number inside the brackets {n} refers to the variables listed after the quotes. The variables are referenced as 0 …. 1…. 2 …3 etc.
27	Readkey ( ) serves only to keep the screen visible until the user presses a key.
28	Ends the subroutine. And the program.

Example Screen

```
Enter Number of Seconds : 4000
----------------------------
Hours   : 1
Minutes : 6
Seconds: 40

:1:6:40
```

Test the Program

Input	Expected Output		
Seconds	**Hours**	**Minutes**	**Seconds**
4000	1	6	40

Order of Precedence

The term *order of precedence* applies to the order in which the operators in an arithmetic expression are used. For example, to evaluate the expression

x = 3 + 4 * 2

Is the answer 14 or 11 ….. The correct answer is 11.

Because multiplication takes precedence over addition.
The rules are (BODMAS):

B -rackets
O
D -ivision
M –ultiplication
A –ddition
S -ubtraction

So x = 3 + 4 * 2
= 3 + 8
= 11

In VB, the operators *, / and MOD have equal precedence; this means that in an expression involving two or more of them, they are simply used in the order that they appear in the expression. These four operators all have higher precedence than + and -. Again, + and - have the same precedence.

Exercises

a) Write a program that calculates the change that should be given to a customer.
The operator enters the total cost of the items purchased and the amount of money the customer tenders. The program then prints the change that that should be returned to the customer.

Total Cost of Items : 8.75
Money Tendered :10.00
Change :1.25

b) Degrees Fahrenheit can be changed to Celsius using the formula: C = (F - 32) * 5/9
ie Celcius = (Fahreneit - 32) * 5 **div** 9 {**div** is integer division})

Write a program to input a temperature in degree Fahrenheit and print out the equivalent Celsius temperature using appropriate messages.

c) Degree Celsius can be changed to Fahrenheit using the formula: F = (C * 9/5) + 32

Write a program to input a temperature in degree Celsius and print out the equivalent Fahrenheit temperature using appropriate messages

d) The above programs only work with integers. Entering a decimal value gives an error.
Modify the above program to accept decimal values.
Hint: declare the **var**iables as type Decimal. Use / (decimal division) instead of **div**.