Introducing VB data types, variables, constants, VB controls and arithmetic

Visual Basic Data Types

Data type	Prefix	Size	Values
Byte	byt	1 byte	positive integer value from 0 to 255
Short	shr	2 byte	integer from –32,768 to +32,767
Integer	int	4 byte	integer from +/- 2,147,483,647
Long	lng	8 byte	integer from +/- 9,223,372,036,854,775,807
Single	sng	4 byte	single-precision, floating-point number
Double	dbl	8 byte	double-precision, floating-point number
Decimal	dec	16 byte	number with up to 28 significant digits
Char	chr	2 byte	Any single character
Boolean	bln	2 byte	True or False
String	str	(4 byte)	Text - Any number/combination of characters
Date	dtm	8 byte	8 character date: #dd/mm/yyyy#
Object	obj	(4 byte)	An address that refers to an object

Variables

- A storage location in memory (RAM)
  - Holds data/information while the program is running
  - These storage locations can be referred to by their names
- Every variable has three properties:
  - **Name** - reference to the location - cannot be changed
  - **Value** - the information that is stored - can be changed during program execution, hence the name "variable"
  - **Data Type** - the type of information that can be stored - can be transfered

Variable Names

- Naming Rules Conventions

- First character must be a letter or underscore
- Must contain only letters, numbers, and underscores (no spaces, periods, etc.)
- Can have up to 255 characters
- Cannot be a VB language keyword

- Naming Conventions
  - Should be meaningful
  - Follow 3 char prefix style - 1st 3 letters in lowercase to indicate the data type
  - After that, capitalize the first letter of each word
  - Example: intTestScore

Declaring a Variable

- A variable declaration is a statement that creates a variable in memory
- Syntax: **Dim *VariableName* As *DataType***
  - Dim (short for Declaration in Memory) - keyword
  - VariableName – The name used to refer to variable
  - As - keyword
  - DataType - one of the many possible keywords to indicate the type of value the variable will contain
- Example: **Dim intLength as Integer**

Declaring and Initializing a Variable

- A starting or initialization value **may** be specified with the Dim statement
- Good practice to set an initial value unless assigning a value prior to using the variable
- Syntax:

  Dim *VariableName* As *DataType* = *Value*
  - Just append " = value" to the Dim statement
  - = 5 → assigning a beginning value to the variable
- Example: **Dim intLength as Integer = 5**

Variable Declaration Rules

- ***Variables MUST be declared prior to the code where they are used***
- ***Declaring an initial value of the variable in the declaration statement is optional***

Default Values for Data Types

Data type value	Default (Initial)
All numeric types	Zero (0)
Boolean	False
Char	“0”
String or Object	Empty
Date	12:00 a.m. on January 1, 0001

Constants in VB

Syntax: **Const *CONST_NAME* As *DataType* = *Value***

Looks like a normal declaration except:

1. Const used instead of Dim
2. An initialization value is required
3. By convention, entire name capitalized with underscore characters to separate words

Assignment Statement

- Syntax: ***variablename = expression***

- Assigns the value of the expression to the variable. (The variable must be on the left and the expression on the right.)
- Example:
  - intNumber1 = 4
  - intNumber2 = 3 * (2 + 2)
  - intNumber3 = intNumber1
  - IntNumber1 = intNumber1 + 6

Explicit Type Conversions

- VB provides a set of functions that perform data type conversions
- These functions will accept a literal, variable name, or arithmetic expression
- The following narrowing conversions require an explicit type conversion
  - Double to Single
  - Single to Integer
  - Long to Integer
- Boolean, Date, Object, String, and numeric types represent different sorts of values and require conversion functions as well

The Val Function

- The *Val function* is a more forgiving means of performing string to numeric conversions
- Uses the form Val(string)
- If the initial characters form a numeric value, the Val function will return that
- Otherwise, it will return a value of zero

The Val Function

Val Function	Value Returned
Val("34.90")	34.9
Val("86abc")	86
Val("$24.95")	0
Val("3,789")	3
Val("")	0
Val("x29")	0
Val("47%")	47
Val("Geraldine")	0

The ToString Method

- Returns a string representation of the value in the variable calling the method
- Every VB data type has a ***ToString*** method
- Uses the form VariableName.ToString
- For example

```
Dim number as Integer = 123
lblNumber.text = number.ToString
```

  - Assigns the string "123" to the text property of the lblNumber control

Performing Calculations with Variables

- ***Arithmetic Operators***

^	Exponent
*	Multiplication
/	Floating Point Division
\	Integer Division
MOD	Modulus (remainder from division)
+	Addition
–	Subtraction
&	String Concatenation (putting them together)

Integer Division Operator

- The backslash (\) is used as an *integer division* operator
- The result is always an integer, created by discarding any remainder from the division
- ***Example***
  - `intResult = 7 \ 2` `'result is 3`
  - `shrHundreds = 157 \ 100` `'result is 1`
  - `shrTens = (157 - 157 \ 100 * 100) \ 10`
    `'result is ?`

Special Mod Operator

- This operator can be used in place of the backslash operator to give the remainder of a division operation

```
intRemainder = 17 MOD 3`result is 2
dblRemainder = 17.5 MOD 3    `result is 2.5
```

- Any attempt to use of the \ or MOD operator to perform integer division by zero causes a *DivideByZeroException* runtime error

Concatenating Strings

- Concatenate: connect strings together
- Concatenation operator: the ampersand (&)
- Include a space before and after the & operator
- Numbers after & operator are converted to strings
- How to concatenate character strings
  - strFName = "Bob"
  - strLName = "Smith"
  - strName = strFName & " “ → “Bob ”
  - strName = strName & strLName → “Bob Smith”
  - intX = 1 intY = 2
  - intResult = intX + intY
  - strOutput = intX & “ + “ & intY & “ = “ & intResult → “1 + 2 = 3”

Self Assignment Operators

- Often need to change the value in a variable and assign the result back to that variable
- For example: var = var – 5
- Subtracts 5 from the value stored in var

Operator	Usage	Equivalent to	Effect
+=	x += 2	x = x + 2	Add to
-=	x -= 5	x = x – 5	Subtract from
*=	x *= 10	x = x * 10	Multiply by
/=	x /= y	x = x / y	Divide by
\=	x \= y	x = x \ y	Int Divide by
&=	x &= “.”	x = x & “.”	Concatenate

Arithmetic Operator Precedence

- Operator *precedence* tells us the order in which operations are performed
- From highest to lowest precedence:
  - Exponentiation (^)
  - Multiplicative (* and /)
  - Integer Division (\)
  - Modulus (MOD)
  - Additive (+ and -)
- Parentheses override the order of precedence
- Where precedence is the same, operations occur from left to right

All Operators Precedence

- Parenthesis
- Exponential
- Multiplication / Division
- Integer Division
- MOD
- Addition / Subtraction
- String Concatenation
- Relational Operators (< , > , >= , <= , <>)
- Logical Operators (AND, OR, NOT)

Precedence Examples

6 * 2 ^ 3 + 4 / 2 = 50
7 * 4 / 2 – 6 = 8
5 * (4 + 3) – 15 Mod 2 = 34

```
intX = 10
intY = 5
intResultA = intX + intY * 5 'iResultA is 35
iResultB = (intX + intY) * 5 'iResultB is 75
dResultA = intX - intY * 5    'dResultA is -15
dResultB = (intX - intY) * 5 'dResultB is 25
```