Chapter 2 Writing Software

❖ Identifiers

Identifiers are names given to namespaces, types (enumerations, structures, classes, standard modules, etc.), type members (methods, events, constants, and properties), and variables. Identifiers must begin with:

- An alphabetic or underscore character (_).
- May be of any length
- After the first character must consist of only alphanumeric and underscore characters.
- Namespace declarations may be declared either with identifiers or qualified identifiers. Qualified identifiers consist of two or more identifiers connected with the dot character (.). Only namespace declarations may use qualified identifiers.
- It should not be a reserved keyword. Ordinarily, identifiers may not match Visual Basic .NET keywords. If it is necessary to declare or use an identifier that matches a keyword, the identifier must be enclosed in square brackets ([]). For example :

Public Class [Public]

❖ Literals

Literals are representations of values within the text of a program. For example,**" x = y * 10"** where 10 is a literal, but x and y are not.

There are many types of literals.

➢ Numeric Literals

Represents any number within specific range of either ***Integer Literals*** (like: byte, short integer, or long integer) by default Visual Basic .NET interprets integer literals as type of integer, or ***Floating Point Literals*** (like: Single, Double, or Decimal). By default Visual Basic .NET interprets floating point literals as type of Double.

➢ String Literals

Literals of type **String** consist of characters enclosed within quotation-mark (**"**) characters. For example, in the following line of code, "hello world" is a literal of type String:

➢ Character Literals

Visual Basic .NET's **Char type** represents a single character. This is not the same as a one-character string; Strings and Chars are distinct types. Literals of type Char consist of a single character enclosed within quotation-mark characters.

➢ Date Literals

Literals of type Date are formed by enclosing a date/time string within number-sign characters. For example:

Dim MyDate As Date MyDate = #11/15/2001 3:00:00 PM#

Date literals in Visual Basic .NET code must be in the format m/d/yyyy, regardless of the regional settings of the computer on which the code is written.

➢ Boolean Literals

The keywords **True** and **False** are the only Boolean literals. They represent the true and false Boolean states. For example:

Dim MyBoolean As Boolean MyBoolean = True

➢ Nothing

There is one literal that has no type: the **keyword Nothing**. Nothing is a special symbol that represents an uninitialized value of any type. It can be assigned to any variable and passed in any parameter. When used in place of a value type, it represents an empty value of that type. For numeric types, this is 0. For the String type, this is the empty string (""). For the Boolean type, this is False.

❖ Variables and Constants

In VB as in other programming language, a program consists of **Instructions** that tells the computer what to do and **Data** that the program uses when it is running. The Data consists of **Constants** or *fixed values that never change* and **Variable** *store values during a program's execution*. Usually, both constants and variables are defined as certain **Data Types**. A variable has **Name** and **Value** which can be change depending on conditions or on information passed to the program**.** When you declaring variable you should be aware of naming conventions as mentioned previously.

➢ Types of Variables

VB.Net provides a wide range of data types. The following table shows the fundamental VB.NET types:

Data Type	Storage Allocation	Value Range
Boolean		**True** or **False**
Byte	1 byte	0 through 255 (unsigned)
Char	2 bytes	0 through 65535 (unsigned)

Date	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
Decimal	16 bytes	0 through +/- 79,228,162,514,264,337,593,543,950,335 (+/-7.9...E+28) with no decimal point; 0 through +/- 7.9228162514264337593543950335 with 28 places to the right of the decimal
Double	8 bytes	-1.79769313486231570E+308 through -4.94065645841246544E-324, for negative values 4.94065645841246544E-324 through 1.79769313486231570E+308, for positive values
Integer	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
Long	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807(signed)
Short	2 bytes	-32,768 through 32,767 (signed)
Single	4 bytes	-3.4028235E+38 through -1.401298E-45 for negative values; 1.401298E-45 through 3.4028235E+38 for positive values
String	Depends on implementing platform	0 to approximately 2 billion Unicode characters

➢ Variable Declaration

The Variable declaration involves giving the variable a name and defining the data type to which it belongs. **Dim** statement is used for variable declaration and storage allocation for one or more variables. Dim is used at module, class, structure, procedure, or block level. You can use the word Dim with one of the following options:

[Shared] [Static] [ReadOnly] Dim [withEvents]

Where,

Shared: declares a shared variable, which is not associated with any specific instance of class or structure. **Optional**
Static: Variable will retain its value, even when the procedure terminated. **Optional**
ReadOnly: Variable can be read but not written. **Optional**
WithEvents: Variable is used to respond to events raised by the instance assigned to the variable. **Optional**
For declaring variables use the following syntax:

Dim Variable-Name boundslist As New data type = initializer

Where,
Variable Name: is name of variable.
Boundlist: [optional]. Is provides list of bounds of each dimension of an array variable.
New: [optional]. It creates a new instance of the class when the Dim statement runs.
Data type: Required if option strict is on. It specifies the data type of the variable.
Initializer: [optional if New is not specified]. It represents the value that assigned to the variable when it is created.
Example:

Static Dim x As Integer

In the above example, 'x' is the variable name while Integer is the data type to which variable x belongs.
Vb.Net also allows defining other value type of variable like **Enum** and reference types of variables like **class.**

➢ Constant Declaration

The constants refer to the fixed values that the program may not alter during its execution that means can't modify after definition. These fixed values are also called **Literals.** Constant can be of any of the basic data types like an integer, double, char or string literals. For declaring constants use the following syntax:

Const ConstName As DataType = Initializer

Where
ConstName: Name of constant.
DataType: Type of constant.
Initializer: The value assigned to the constant.
Example:

Public Const message1 As String = "Hello…"