Unit I
VB.NET Syntax: Fundamentals and Data Storage

Syntax Fundamentals

- VB.NET program consists of statements, or instructions

- What are statements or instruction?

  They tell the operating system and computer hardware to carry out some task

- In your source code, statements are written one by line. Two or more statements can appear in a line separated with : (not recommended)

- By default, the statements are executed or carried out , one at a time, from top to bottom

Source code rules:

- Some **fundamental rules** regarding source code
  - No special character is used to mark the end of the line. To start a new line, simply press Enter
  - Blank lines and indentation are ignored.
  - Capitalization does not matter. You'll see, however, that the Visual studio editor enforces certain capitalization guidelines

Components of a statement

- A statements can consists of :
  - Keywords: reserved words having special meaning in Visual Basic
  - Operators: symbols used to perform operations
  - Variables: symbolic names given to values stored in memory
  - Literal Values: simple values, for example, 5 or “Hello”
  - Constants: same as variables except their assigned values cannot change
  - Expression: is a combination of any of the above terms, and yields a value

Comments

- A comment is a section of Visual Basic code which is ignored by the compiler
- This is used by programmers to document the operationof their source code
- In Visual Basic, there are two styles of comments
  - Any line that starts with the keyword Rem is a comment. Example:

    REM Program to add two numbers
  - Any text that follows a single quote(‘) character is a comment.
    Example:

    ‘ Program to add two numbers

Data Storage

- Almost every program works with data, or information, of one type or another
- In programs, we store information in *variables and constants*
- A *variable* can hold information which can change during program execution, unlike *constants*
- What is a variable?
  - A variable is a *named location* where you *can store data*.
  - They are *stored in RAM* while program is running
  - We *refer to the variable by its name* in the program code when we need to utilize it
  - It also has a *type*. This determines what type of data the variable can store

Visual Basic.NET data types

Data type	Storage Space (in Bytes)	Values
Boolean	4	True or False
Byte	1	Integers 0 to 255
Char	2	Integers 0 to 65535
Date	8	Date/Time values between January 1, 0001 and December 31, 9999
Decimal	12	Floating point values with 29 digits, with anywhere from 0 to 28 of the digits to the right of the decimal point. Range approximately +/- 7.9E28
Double	8	Floating point values approximately - 17.9E308 to-4.9E-324 for negative values. For positive values 4.9E-324 to 1.7E308

Visual Basic.NET data types

Data type	Storage Space (in Bytes)	Values
Integer	4	Integers -2,147,483,648 to 2,147,483,647
Long	8	Integers approximately -9E18 to 9E18
Object	4	Object references or any other data type
Short	2	Integers -32,768 to 32,767
Single	4	Floating point values. Approximately, -3.4E38 to -1.4E-45 for negative values, 1.4E-45 to3.4E38 for positive values
String	Twice the number of characters plus 10	Any text from 0 to approximately 2 billion characters in length.

Variable Names

- Every variable must have a name which is unique within its scope
- Naming rules:
  - The maximum length is *255 characters*
  - The *first character must be a letter*, followed by any other character, numbers or letters or an underscore
  - Names are *not case sensitive,* meaning there is no difference between the name, *num and Num, (considered to be the same variable)*
  - *VB.NET keywords cannot be used*
- An advice: Give meaning full names to your variable to describe what they hold. Increase in readability and understandability of source code

Declaring variables

- Each variable must be declared in the program before it can be used
- When we declare, we are setting aside storage space for it
- How much space?
  - The space kept aside depends on the data type of the variable
- We can also initialize variable during the time of declaration
- To declare variables, we use the *Dim* keyword
- Syntax:

  Dim <variablename> As <datatype> [=value]
- Here [=value] is optional.
- Example: Dim num As Integer

  Dim num As Integer = 10

Variable declaration

- Declaring many variables of the same type in one Dim statement
  - Syntax:

    Dim <variable1>,<variable2>,…,<variableN> As <datatype>
  - Example: Dim num1, num2 As Integer
- Declaring variables of different types in a single Dim statement, as follows

  Dim <variable1> As <datatype1> ,<variable2> As <datatype2>,…,<variableN> As <datatypeN>
  - Example:

    Dim rollNo as Integer, name As String

Where to write a declare statement?

- Usually, at the beginning of a module, procedure, functions etc.

Default Initial values of variables

- If no initial value is provided to a variable during declaring, what value will it initially hold?

Variable Type	Default Intial Value
Any numeric type	0
String	An empty string
Object	The special value Nothing
Boolean	False
Date	Midnight on 01/01/01

Constants

- A constant is a piece of program data that cannot change during program execution.
- The values of constants are set when the source code is written
- There are two types of constants
  - Literal Constants: These are nothing more than values that we can give to our variables. Can be of any type. Example: Integer literal = 123, String literal = “Hello”, Double literal = 8.96, etc
  - Symbolic Constants

Symbolic Constants

- Such types of constants are declared before they can be used, with the *Const keyword*. Syntax:

  Const <CONSTNAME> As <datatype> = value

- An advice: constant names are usually declared in capital letters to distinguish them from variables
- Example:

  Const MAXIMUM As Integer = 30

- How to declare the constant ∏ (pi) in VB.NET?

  Ans: Const PI As Single = 3.14

  or, Const PI As Double = 3.14

Learning about Data types

About Single & Double data types

- Double-precision numbers are stored internally with *greater accuracy* than single-precision numbers
- Thus with more precision we require *more memory*
- Single-precision provides 7 significant digits whereas double precision provides 14 significant digits approximately
- For example:

```
Dim a As Single, b As Double
a = 1 / 3
b = 1 / 3
```

- We will have:

```
a=0.3333333
b=0.333333333333333
```

- Hence, b-a *will not be zero*!

About Decimal Data type

- The Decimal data type are stored internally as *Integers* in *12 bytes* and are *scaled by a power of 10.*
- *Scaling power:* determines the number of decimal digits to the right of the floating point
  - It is an integer value from 0 to 28.
  - If scaling power=0, then 10^0 means no decimal digits
  - If scaling power=0, then 10^{28} means 28 decimal digits
- The Decimal data type is new to VB.NET and has replaced the *Currency data type* of previous versions of VB.

About Decimal Data type

- When using Decimal numbers, VB keeps track of the decimal digits and treats all values as *integers.*
- Example:

  Dim dec_var As Decimal = 235.85
- Memory Representation
  - This declaration will set aside 12 bytes in memory for dec_var
  - VB.NET will take the value 235.85 and store it as integer value 23585
  - We know that, 235.85*100=23585, thus scaling power = 2, 10^2=100
  - First, VB.NET multiplies the value by 100 to make it an integer.
  - Then, it divides by 100 to restore the original number.

Boolean

- The Boolean data type stores *True / False* values.
- Boolean variables are, in essence, *integers* that take the value 1 (for True) and 0 (for False). Actually, *any non-zero value* is considered True.
- Boolean variables are declared as:

  Dim chk As Boolean
- By default, they are initialized to *False*

String

- The String data type *stores only text,* basically anything with double quotes(“”) is taken to be a string.

- String variables are declared as follows:

  Dim str As String

- We can store nearly 2 GB of text in a string variable (that’s 2 billion characters)
- Example:

  str = “Hello, how are you?”
  str = “25,000”
  str = “” ‘empty string

Character

- Character variables *store a single Unicode character* in two bytes.
- VB.NET characters are unsigned short integers (UInt16).
  - Example:

    Dim ch1 As Char = “a”, ch2 As Char = “ABC”
  - In the above case, ch1 will store “a”
  - ch2 will store “A”, i.e., only the first character of the string is assigned to the variable.
- The integer values corresponding to the English characters are the ANSI codes of the equivalent characters (“A” is 65 & “a” is 97)

Date

- Date and Time values are stored internally in a specific format.
- They are *double-precision numbers*
  - The *integer part* represents the *date*
  - The *fractional part* represents the *time*.
- A variable declared as Date can store both *date and time* values.
- # or “” symbols are used to assign a particular date and time format to Date variable
- Example:

```
Dim d As Date
d = #01/01/2004#
d = #08/27/2008 08:10:25 PM#
d = “June 9, 2007”
d = Now( )
```

- The Now( ) function returns the *current* date and time.

Object

- VB.NET supports *variants.* These are variables without a fixed data type
- Variants *can store all types of values*, from a single-precision character to an object.
- To *declare* a variant using the Object data type:

```
Dim var As Object
```

- Here, var="Welcome"
  var=3.14
  var=2
  are all acceptable

Special Values

Infinity & NaN

- VB.NET can represent two very special values,
- These values are not numeric but are produced by numeric calculations
- They are
  - Infinity :
  - NaN (not a number)

Infinity

- Unfortunately, computers cannot represent infinity, so they produce an error.
- If an error is encountered, the program will stop execution.
- But, in VB.NET if a calculation leads to infinity, then it will display the word, Infinity
- Example:

```
Dim a, b As Double
b = 30
a = b / 0
Msgbox(a)
```

- What will be displayed?
  - Other languages may provide error messages but VB.NET will show, Infinity

NaN(Not a Number)

- The value NaN indicates that the *result of an operation cannot be defined*, that is, it is *not a regular number*, not zero, and not Infinity.
- Example:

```
Dim a, b, c As Double
a = 0
b = 0
c = a / b
Msgbox(c)
```

- What will be displayed?
  - Other languages may provide error messages but VB.NET will show, NaN

Nothing Value

- The Nothing value is useful with Object variables and indicates that a variable that has not been initialized.
- *Example*:

```
Dim a As Object
a = 5000 ‘initialized to the value 5000
a = Nothing ‘to remove any initializations
```