I101/B100
Problem Solving with Computers

By:

Dr. Hossein Hakimzadeh
Computer Science and Informatics
IU South Bend
What do we need to learn in order to write computer programs?

- Fundamental programming constructs:
  - Variables,
  - Arithmetic operators,
  - Input and output
  - Conditions,
  - Loops,
  - Procedures and functions,
  - Arrays,
  - Structures, classes and objects,
  - Files
Variables:

- Remember this figure.

- Variables in VB are the method by which programmers can access and manipulate memory.
Variables

- A variable is the **name given to a memory location**. (the memory location may hold data items such as numbers, characters, etc.)

- VB programs **store and manipulate their data by using variables**.

**Syntax:**
- Dim variable_name AS data_type

**Example:**

**Declaring Variables:**
- Dim x As Integer
- Dim y As Double
- Dim z As char

**Using the Variables:**
- x = 5
- y = 4.8
- z = ‘A’
Assignment Statement

- **Syntax:**
  - `variable = expression`
  - "=" is the assignment operator
  - An expression is a combination of variables, constants, numbers and operators.

- **Examples:**
  - `N1 = N2`  `variable = variable`
  - `N1 = 5`  `variable = constant number`
  - `N1 = N2 * 5 +1`  `variable = expression`
  - `N1 = N1 + 5`
Data Types

In VB, each variable must have a **data type**. VB provides a number of built-in data types. These include:

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer</td>
<td>4 bytes</td>
<td>integer</td>
</tr>
<tr>
<td>Double</td>
<td>8 bytes</td>
<td>floating point or real number</td>
</tr>
<tr>
<td>Decimal</td>
<td>12 bytes</td>
<td>Similar to float but more exact</td>
</tr>
<tr>
<td>Long</td>
<td>8 bytes</td>
<td>Long integer (64 bit integer)</td>
</tr>
<tr>
<td>Char</td>
<td>2 bytes</td>
<td>Unicode character</td>
</tr>
<tr>
<td>Byte</td>
<td>1 byte</td>
<td>0 to 255</td>
</tr>
<tr>
<td>Boolean</td>
<td>1 byte</td>
<td>True or False</td>
</tr>
<tr>
<td>Date</td>
<td>8 bytes</td>
<td>holds time and date</td>
</tr>
<tr>
<td>String</td>
<td>variable</td>
<td>holds variable sized strings</td>
</tr>
</tbody>
</table>
Declaring a Variable

- In most modern programming languages **variables must be declared** before they are used.

- Examples of variable declaration in VB:
  - Dim X As Integer
    - Variable name: X
    - Data type: Integer
Declaring a variable will accomplish the following:

1) Associate the variable name with some memory location in which the data will be stored.

2) Tell the compiler how much memory is needed for each variable:
   
<table>
<thead>
<tr>
<th>Variable</th>
<th>Memory Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>4 bytes</td>
</tr>
<tr>
<td>y</td>
<td>8 bytes</td>
</tr>
<tr>
<td>c</td>
<td>2 bytes</td>
</tr>
</tbody>
</table>

   `Dim X As Integer`
   `Dim Y As Double`
   `Dim C As Char`

3) Allow the compiler to perform type compatibility checking. For example, the compiler can easily identify problems like:

   c = 5.8;  `compiler error` (c is a character not a floating point number)
   y = ‘hello’  `compiler error` (Y is double not a string)
Type Checking

Traditionally the BASIC language was not a strongly typed language. However, in VB .Net the programmer can explicitly tell the compiler to enforce strong type checking.

Strong type checking can be enforced with the help of the following two statements.

- **Option Strict On**  
  'Tells the compiler to enforce strict type checking.

- **Option Explicit On**  
  'Tells the compiler that variables must be declared before use.
The numbers **10** and **10.0** are not the same and therefore, have different types:

- **10** is an integer number.
- **10.0** is a floating point number.

The compiler will **automatically convert** an integer number to its equivalent **float** number however, **the reverse is not true**.

**Example:**

- Dim x as double
  - x = 10  `Will automatically convert to a float`
- Dim y as integer
  - y = 10.5  `Error or warning in most compilers`
Arithmetic Operators:

- Remember this figure.

- **Arithmetic operators** are implemented in the Arithmetic Logic Unit of the computer.
Arithmetic Operators

- Addition
- Subtraction
- Multiplication
- Real division
- Integer division
- Remainder of an integer division (modula)

Example:

- 5.0 / 2.0  ==> 2.5
- 5 \ 2       ==> 2
- 4 / 2       ==> 2
- 5 MOD 2    ==> 1
Arithmetic Operators

Module Module1
Sub Main()

Dim x, y As Integer
x = 5
y = 2
Console.WriteLine(x / y) ' output = 2.5
Console.WriteLine(x \ y) ' output = 2
Console.WriteLine(x Mod y) ' output = 1
Console.WriteLine(y Mod x) ' output = 2

Dim a, b As Double
a = 5.0
b = 2.0
Console.WriteLine(a / b) ' output = 2.5
Console.ReadLine() ' Pause!
End Sub
End Module
Input and Output

- I/O is an essential part of any application.
**Output (Console Applications)**

- **Output** can be sent to the computer display by using the **Write() or WriteLine()** procedures.

  **Syntax:**

  ```csharp
  System.Console.WriteLine(argument)
  ```

  Arguments can be:

  - **constant** *(string or numeric)*
  - **variable**
Example 1:

```csharp
Option Strict On
Option Explicit On

Module Module1
    Sub Main()

        System.Console.WriteLine(3)
        System.Console.WriteLine(3 * 5)
        System.Console.WriteLine("HELLO")

    End Sub
End Module
```

3
15
HELLO
Example 2:

```vbnet
Option Strict On
Option Explicit On
Module Module1
    Sub Main()
        Dim temp As Integer = 75
        Console.WriteLine("The temperature is {0} degrees", temp)

        Dim A, B As Integer
        A = 10
        B = 20
        Console.WriteLine("A is = {0} and B is = {1}", A, B)
    End Sub
End Module
```

The temperature is 75 degrees
A is = 10 and B is = 20
Input (Console Applications)

- **Input** can be read from the keyboard by using the `ReadLine()` function.

- Syntax:
  - `Variable = Console.ReadLine()`

- Note that `ReadLine()` will return a **STRING** as the result. So, if a value other than a string is desired the programmer must perform the conversion.
Input (Console Applications)

- Option Strict On
- Option Explicit On

Module Module1

Sub Main()

Dim Name As String
Console.Write("What is your name? ")
Name = Console.ReadLine()
Console.WriteLine("Good day {0}.", Name)

Dim Age As Double
Console.Write("How old are you? ")
Age = CDbl(Console.ReadLine())
Console.WriteLine("{0} you are approximately {1} days old.", Name, Age * 365)

Console.ReadLine() ' Pause

End Sub

End Module
Output (Windows Applications)

- Displays a message box that can contain text, buttons, and symbols that inform and instruct the user.

- `MessageBox.Show("Hello")`

- `MessageBox.Show("Hello", "Dialog Box Caption", MessageBoxButtons.OK, MessageBoxIcon.Exclamation)`
Input (Windows Applications)

- Displays a prompt in a dialog box, waits for the user to input text or click a button, and then returns a string containing the contents of the text box.

- Dim strName As String
  - strName = InputBox("What is your name?", "Name")

- MessageBox.Show(strName)
Reminder: Errors Encountered by Programmers

- Syntax Errors:
  - When VB’s rules for punctuation, format, or spelling is violated.
  - Most Syntax errors are detected by the editor in the IDE.

- Runtime Errors
  - If your program halts or crashes during execution.
  - Example:
    - Divide by zero.
    - Finding the square root of a negative number.
    - Trying to read from a non-existing file.
  - Runtime errors are also known as ‘EXCEPTIONS’

- Logic Errors
  - The program runs but produces incorrect results.