Introduction

- Computer Memory & Variables
- Data Types
- Variable Names & Declaration
- Assignment Statements
- Arithmetic Operators & Math Functions
- String Variables
Computer Memory

• Values in programs are stored in memory at locations with an address for each location

Variables

• Names used in place of addresses to represent information in a program
• Can contain numbers and/or characters
**Data Types**

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Storage size</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>1 byte</td>
<td>0 – 255</td>
</tr>
<tr>
<td>Integer</td>
<td>2 bytes</td>
<td>-32,767 → +32,767</td>
</tr>
<tr>
<td>Long (integer)</td>
<td>4 bytes</td>
<td>- Big # → + Big #</td>
</tr>
<tr>
<td>Single (real)</td>
<td>4 bytes</td>
<td>-1x10^{38} → +1x10^{38}</td>
</tr>
<tr>
<td>Double (real)</td>
<td>8 bytes</td>
<td>- Big # → + Big #</td>
</tr>
<tr>
<td>String</td>
<td>1 byte / char</td>
<td>Large</td>
</tr>
</tbody>
</table>

**Naming Variables**

- **Type** and **Name** of every variable must **declared**
- **Rules** for declaring variables
  - Start with a letter (a – z)
  - Contain only:
    - letters (a – z), digits (0 – 9) and underscore (_)
  - VB is **not** case sensitive

<table>
<thead>
<tr>
<th>x</th>
<th>y12</th>
<th>sum_1</th>
<th>austinTemp</th>
</tr>
</thead>
<tbody>
<tr>
<td>radius</td>
<td>area</td>
<td>tax_rate</td>
<td>TABLE</td>
</tr>
</tbody>
</table>
Declaring Variables

- Variables are defined (declared) with a Declaration Statement
- ALWAYS declare variables

**General form:**

```
Dim variable_name as data-type
```

**Examples:**

```
Dim num1 as Integer
Dim num2 as Single, sum as Single
Dim radius as Double
```

Assigning Values to Variables

- **General form:** `identifier = expression`
- **Examples:**

```
Dim sum As double
sum = 0.0
```

```
Dim x As integer
Dim y As integer
Dim z As integer
x=0
y=0
z=2
y=z
```

```
Sometimes in VB

ObjectName.PropertyName = expression

Form1.Caption = “MyForm”
```
Initialization

• Numeric variables are automatically initialized to 0:
  − Dim varName As Double

• To specify a nonzero initial value
  − Dim varName As Double = 50

Assignment Statement: Example

Dim numVar1 As Double = 5
Dim numVar2 As Double = 4
numVar1 = 3 * numVar2
lstBox.Items.Add(numVar1)

Output: 12
Multiple Declarations

Dim a, b As Double

Two other types of multiple-declaration statements are

Dim a As Double, b As Integer
Dim c As Double = 2, b As Integer = 5

Arithmetic Operators

• Exponentiation ^
• Negation -
• Multiplication *
• Division /
• Addition +
• Subtraction -
• Integer division \n• Modulo division Mod
Displaying Numbers

• The statement
  
  `lstBox.Items.Add(n)`
• displays the value of `n` in the list box.
• Expressions can also be used
  
  `Dim balance As Double = 1000`
  
  `lstBox.Items.Add(1.05 * balance)`

Output: 1050

Displaying Numbers

• How can I show a number on the screen?
• Use a “List Box”
  
  `ListBox1.Items.Add(n)`

`List Box named “ListBox1”`
Example

```
Private Sub Button1_Click(ByVal sender As System.Object)
    ListBox1.Items.Clear()
    ListBox1.Items.Add(n + 2)
    ListBox1.Items.Add(n - 2)
    ListBox1.Items.Add(n * 2)
    ListBox1.Items.Add(n / 2)
    ListBox1.Items.Add(n ^ 2)
    ListBox1.Items.Add(2 * (n + 4))
End Sub
```

ListBox1.Items.Add(n)

“Run” and get display

Example Using Variables

```
Private Sub Button1_Click(ByVal sender As System.Object)
    Dim speed As Double
    Dim distance As Double
    ListBox1.Items.Clear()
    speed = 50
    timeElapsed = 14
    distance = speed * timeElapsed
    ListBox1.Items.Add(distance)
    distance = 410
    timeElapsed = distance / speed
    ListBox1.Items.Add(timeElapsed)
End Sub
```

```
703
82
```
Constants

- **Numeric constant**: numeric data, e.g. 7
- **String constant**: character data, e.g., “the”
- **Hardcoded**: constants are programmed directly into source code
- **Named constants**: constants are defined by names and can be changed easily

**Example**

```
Const constantName [As DataType] = Expression

Const PI As Double = 3.14159
```

Modulo Division

- **Modulus operator**:
  - The result is the remainder when the first argument is divided by the second

```
8 6 5 0 2
6\ 50 7\ 43 12\ 60 17\ 4 2\ 5
48 42 60 0 4
2 1 0 4
R = 2 R = 1 R = 0 R = 4 R = 1
50 Mod 6 = 2 43 Mod 7 = 1 60 Mod 12 = 0 4 Mod 17 = 4
```
**Integer Division**

- Division between two integers results in an integer.
- The result is truncated, not rounded
- Example:
  
  \[
  \frac{5}{3} = 1 \\
  \frac{3}{6} = 0
  \]

**Examples**

```
Dim a as Integer = 27
Dim b as Integer = 6
Dim c as Integer

Dim a as Integer
Dim b as Integer
Dim c as Integer

Dim a as Integer
Dim b as Integer
Dim c as Integer

b = 6
c = 17
a = c \ b
a = ?
```

\[
\text{c = 6} \quad \text{a = 2}
\]
Priority of Operators

1. Exponentiation
2. Negation
3. Multiplication & Division (left to right)
4. Integer Division
5. Modulo Division
6. Addition & Subtraction (left to right)
7. String Concatenation

<table>
<thead>
<tr>
<th>Order</th>
<th>Operators</th>
<th>Example</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parentheses ( )</td>
<td>(2 + 3) * 7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>^</td>
<td>2^3 + 1</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>*, /, , Mod</td>
<td>2 + 3 * 7</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>+, -</td>
<td>10 – 4 * 2 + 1</td>
<td></td>
</tr>
</tbody>
</table>

Math Functions

Abs(x)  Absolute value of x
Sqr(x)  Square root of x, where x >= 0
Exp(x)  Computes the value of \( e^x \)
Log(x)  In x, the natural logarithm of x - base e
Sin(x)  sine of x, where x is in radians
Cos(x)  cosine of x, where x is in radians
Tan(x)  tangent of x, where x is in radians
Atn(x)  inverse tangent of x
Three Types of Errors

- Syntax error
- Runtime error
- Logic error

Syntax Errors

- Misspellings
  ```csharp
  lstBox.Itms.Add(3)
  ```
- Omissions
  ```csharp
  lstBox.Items.Add(2 + )
  ```
- Incorrect punctuation
  ```csharp
  Dim m; n As Integer
  ```
Runtime Error

Overflow error

Dim numVar As Integer = 1000000
numVar = numVar * numVar

Logical Error

Dim average As Double
Dim m As Double = 5
Dim n As Double = 10
average = m + n / 2

Value of average will be 10. Should be 7.5.
Error List Window

Dim m; n As Double
lstResults.Items.Add(5)
lstResults.Items.Add(a)

String Literal

A string literal is a sequence of characters surrounded by quotation marks.

Examples:

"hello"
"123-45-6789"
"#ab cde?"
String Variable

A string variable is a name to which a string value can be assigned.

Examples:

- country
- ssn
- word
- firstName

String Variable (continued)

- Declaration:
  
  ```Dim firstName As String```

- Assignment:
  
  ```firstName = "Fred"```
String Variable (continued)

You can declare a string variable and assign it a value at the same time.

```vbs
Dim firstName As String = "Fred"
```

String Variable

You can assign the value of one string variable to another.

```vbs
Dim strVar1 As String = "Hello"
Dim strVar2 As String = "Goodbye"
strVar2 = strVar1
lstOutput.Items.Add(strVar2)
```

Output: Hello
Strings

- Name used to refer to a string
- General form:
  
  ```vba
  Dim VarName As String
  ```

  Text Boxes for Input & Output

  - The contents of a text box is always a string.

  - Input example:
    ```vba
    strVar = txtBox.Text
    ```

  - Output example:
    ```vba
    txtBox.Text = strVar
    ```
Data Conversion

Because the contents of a text box is always a string, sometimes you must convert the input or output.

```vbnet
dblVar = CDb1(txtBox.Text)  ' converts a String to a Double

txtBox.Text = CStr(numVar)  ' converts a number to a string
```

Concatenation

Combining two strings to make a new string

```vbnet
quote1 = "We'll always "
quote2 = "have Paris."
quote = quote1 & quote2
txtOutput.Text = quote & " - Humphrey Bogart"
```

Output:

We'll always have Paris. - Humphrey Bogart
String Properties and Methods

"Visual".Length is 6.

"Visual".ToUpper is VISUAL.

"123 Hike".Length is 8.

"123 Hike".ToLower is 123 hike.

"a" & " bcd ".Trim & "efg" is abcdefg.

Positions in a String

Positions of characters in a string are numbered 0, 1, 2, ....

Consider the string “Visual Basic”.

Position 0: V
Position 1: i
Position 7: B

Substring “al” begins at position 4
**Substrings**

Let \( str \) be a string.

\( str\text{.Substring}(m, n) \) is the substring of length \( n \), beginning at position \( m \) in \( str \).

“Visual Basic”.Substring(2, 3) is “sua”
“Visual Basic”.Substring(0, 1) is “V”

**“IndexOf” Method**

Let \( str1 \) and \( str2 \) be strings.

\( str1\text{.IndexOf}(str2) \)

is the position of the first occurrence of \( str2 \) in \( str1 \).

(Note: Has value -1 if \( str2 \) is not a substring of \( str1 \).)

"Visual Basic".IndexOf("is") is 1.
"Visual Basic".IndexOf("si") is 9.
"Visual Basic".IndexOf("ab") is -1.
Empty String

- "" is a string with no characters - empty string
- `lstBox.Items.Add("")` skips a line in the list box.
- The contents of a text box can be cleared with either the statement
  
  ```csharp
  txtBox.Clear()
  ```
  or the statement
  
  ```csharp
  txtBox.Text = ""
  ```

Incrementing a Variable

<table>
<thead>
<tr>
<th>operator</th>
<th>example</th>
<th>equivalent statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>+=</td>
<td>x += 2;</td>
<td>x = x+2;</td>
</tr>
<tr>
<td>-=</td>
<td>x -= 2;</td>
<td>x = x-2;</td>
</tr>
</tbody>
</table>
Comments

Private Sub btnCompute_Click (...) Handles btnCompute.Click
    'Calculate the balance in an account
    Dim rate As Double 'Annual rate of interest
    Dim curBalance As Double 'Current balance

Internal Documentation

1. Other people can easily understand the program.
2. You can understand the program when you read it later.
3. Long programs are easier to read because the purposes of individual pieces can be determined at a glance.
Line Continuation

A long line of code can be continued on another line by using an underscore (_) preceded by a space

```plaintext
msg = "I'm going to make " & _
      "him an offer he can't refuse."
```

Scope

• The **scope** of a variable is the portion of the program that can refer to it.

• Variables declared inside an event procedure are said to have **local scope** and are only available to the event procedure in which they are declared.
Scope (continued)

• Variables declared outside an event procedure are said to have class-level scope and are available to every event procedure.

• Usually declared after
  
  Public Class formName

  (In Declarations section of Code Editor.)

Summary

• Computer Memory & Variables
• Data Types
• Variable Names & Declaration
• Assignment Statements
• Arithmetic Operators & Math Functions
• String Variables