VB.Net

AS Computer Studies: PROGRAMMING

WORKING WITH DATA

Learning Objectives

- **Declaring Variables**; Input Boxes; Overflow
  - Explain **what a variable is** and what they are used for.
  - Explain **how to declare variables** and explain the necessity for doing so.
  - Explain why **using comments** is useful, how to enter them and what VB does with them.
  - Explain the **Overflow error**.

Variables

- a program runs -> data that it uses is stored in RAM.
- **variable** = memory location in RAM
  - with a name made up by the programmer to identify that address where a particular piece of data is stored.

e.g. I would want to store your test results and calculate an average.

For this I'd need at least two variables:

1. Score
2. Average

...both numbers.

Declaring variables

- Tells the VB compiler two things about a variable:
  - Its name (identifier)
  - Its data type.
- **ALWAYS declare each variable** as certain errors/problems can occur if you don't:
  - **Name-conflict errors** – using a variable name/identifier which VB already uses for its own purposes.
  - **Spelling mistakes** – spelling a variable name/identifier one way the first time you use it and then accidentally spelling another way later in the code.
  - **VB assigns the Object data type if the data type of a variable is not declared** – this is slower to access and uses more memory.

Option Explicit

If On - VB looks for undeclared variables as you type in the code (so forcing you to do so).

If Off – VB does not look for undeclared variables as you type in the code but runtime errors may occur (as described on the previous slide).

In VB2008 it is On by default, but you can check it is on by:

- Tools menu, choose Options.
- Open the Projects and Solutions node.
- Choose VB Defaults.

Conventions for Naming Variables

- Always use **meaningful** names / identifiers.
- You **cannot use spaces**. If two words are needed, best to follow the convention of making each word start with a capital e.g. DateOfPayment
- Consider the data type carefully – the right size, range of values, memory used

Dim

Use this statement to **declare variables**.

Dim (Variable name) As (Data Type)

e.g.

Dim Number **As** Integer

Dim Payment **As** Decimal

Dim DateOfPayment **As** Date

Code Comments

- As programs become more complicated it is useful to be able to write explanations of each line but obviously we want VB to ignore them.

- This is achieved by using apostrophes ‘ .

- **On this course when you write various practice code you will write comments to explain most lines of your code.**

- NOTE: in the summer exam you will lose marks if you don’t include RELEVANT comments (NOT on every line though!!!)

Add two numbers

Specification:

Allow the user to enter two numbers and display the sum.

- Create a new Console application named '**AddTwoNumbers**'.
- Declare three variables needed for this project:
  - FirstNumber
  - SecondNumber
  - Sum

Add two numbers

- For simplicity, assume all three variables are integers.

Module1.vb* | Start Page

Module1

```
1  Module Module1
2
3      Sub Main()
4          Dim FirstNumber As Integer
5          Dim SecondNumber As Integer
6          Dim sum As Integer
7
8      End Sub
9
10 End Module
11
```

- Notice the 3 variables are underlined in green. Why? How did you find out?

Add two numbers

- Now add an input/output statement to ask the user to type in the first number.
  - Hint: use a Console.write or .writeline command
- Once the number is typed in, you need to store it in the first variable.
  - Hint: use a Console.read or .readline command
- Repeat the above for the second number, and add a couple of simple comments.
- Your code should look something like this:

Add two numbers

Module1.vb* | Start Page

Module1 | Mai

```
1  Module Module1
2
3      Sub Main()
4          Dim FirstNumber As Integer
5          Dim SecondNumber As Integer
6          Dim sum As Integer
7
8          'ask the user to enter the numbers, then store those numbers in
9          'the relevant variables
10         Console.Write("Enter the first number: ")
11         FirstNumber = Console.ReadLine()
12         Console.Write("Enter the second number: ")
13         SecondNumber = Console.ReadLine()
14     End Sub
15
16 End Module
```

Add two numbers

...equivalent to this:

t Page / Module1.vb

dule1 | Main

```
1  Module Module1
2
3      Sub Main()
4          Dim FirstNumber As Integer
5          Dim SecondNumber As Integer
6          Dim sum As Integer
7
8          'ask the user to enter the numbers, then store those numbers in
9          'the relevant variables
10         Console.Write("Enter the first number: ") : FirstNumber = Console.ReadLine()
11         Console.Write("Enter the second number: ") : SecondNumber = Console.ReadLine()
12         Console.ReadLine()
13     End Sub
14
15 End Module
```

Add two numbers

- Now add code to confirm to the user what numbers they've entered. For example:

```
Sub Main()
    'declare the variables
    Dim FirstNumber As Integer
    Dim SecondNumber As Integer
    Dim sum As Integer

    'ask the user to enter the numbers, then store those numbers in
    'the relevant variables
    Console.Write("Enter the first number: ") : FirstNumber = Console.ReadLine()
    Console.Write("Enter the second number: ") : SecondNumber = Console.ReadLine()
    Console.ReadLine()

    'confirm the numbers to the user
    Console.WriteLine()
    Console.WriteLine("You've entered the following numbers: " & FirstNumber & " and " & SecondNumber)
    Console.ReadLine()
```

Add two numbers

- Finally, perform the addition and display the result. For example:

```
Sub Main()
    'declare the variables
    Dim FirstNumber As Integer
    Dim SecondNumber As Integer
    Dim sum As Integer

    'ask the user to enter the numbers, then store those numbers in
    'the relevant variables
    Console.Write("Enter the first number: ") : FirstNumber = Console.ReadLine()
    Console.Write("Enter the second number: ") : SecondNumber = Console.ReadLine()
    Console.ReadLine()

    'confirm the numbers to the user
    Console.WriteLine()
    Console.WriteLine("You've entered the following numbers: " & FirstNumber & " and " & SecondNumber)
    Console.ReadLine()

    'add the two numbers
    sum = FirstNumber + SecondNumber
    Console.WriteLine("The sum is: " & sum)
    Console.ReadLine()
End Sub
```

Add two numbers

- Now, comment out the three declaration lines. What do you notice:
  - You will get a blue squiggly line underneath the variable names
- Hover over this and you will get a message saying 'Name 'FirstNumber' is not declared'.
- Why?
- The variable has not been declared!
- As option explicit (mentioned on slide 5) is on, VB **forces** you to **declare your variables**.
- As explained before, it is good programming practice to do so.
- Remove the comment applied above.

Add two numbers

- Next, you will change the **data type** used for the three variables, from Integer to String (which is simple text):

```
1  Module Module1
2
3      Sub Main()
4          'declare the variables
5          Dim FirstNumber As String
6          Dim SecondNumber As String
7          Dim sum As String
8
```

- Run the program. What do you notice? Why do you think that happened?
- Change the data types back to Integer.

Overflow

- Run the program and try entering the number 3000000000
- You will get an **Overflow error**.

- This is caused by the variable being asked to hold a number **outside its Data Type's range**.

- Remember to look at the tables of data types before declaring any variable.

Numeric data types

Type	Storage	Range of Values
Byte	1 byte	0 to 255
Integer	2 bytes	-32,768 to 32,767
Long	4 bytes	-2,147,483,648 to 2,147,483,648
Single	4 bytes	-3.402823E+38 to -1.401298E-45 for negative values 1.401298E-45 to 3.402823E+38 for positive values.
Double	8 bytes	-1.79769313486232e+308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232e+308 for positive values.
Currency	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Decimal	12 bytes	+/- 79,228,162,514,264,337,593,543,950,335 if no decimal is use +/- 7.9228162514264337593543950335 (28 decimal places).

Non-numeric data types

Data Type	Storage	Range
String (fixed length)	Length of string	1 to 65,400 characters
String (variable length)	Length + 10 bytes	0 to 2 billion characters
Date	8 bytes	January 1, 100 to December 31, 9999
Boolean	2 bytes	True or False
Object	4 bytes	Any embedded object
Variant (numeric)	16 bytes	Any value as large as Double
Variant (text)	Length+22 bytes	Same as variable-length string

Learning Objectives - review

- **Declaring Variables**, Input Boxes, Overflow
  - Explain **what a variable is** and what they are used for.
  - Explain **how to declare variables** and explain the necessity for doing so.
  - Explain why **using comments** is useful, how to enter them and what VB does with them.
  - Explain the **Overflow error**.

Required Reading

- Each week you will be given required reading.
- If you fail to do this, you will 100% find the lessons which follow it EXTREMELY difficult.
- Before next lesson you should have read:
- **Programming Guide – READ p10-14**
- **Tasks booklet – DO Tasks 2 and 3, p5-8**