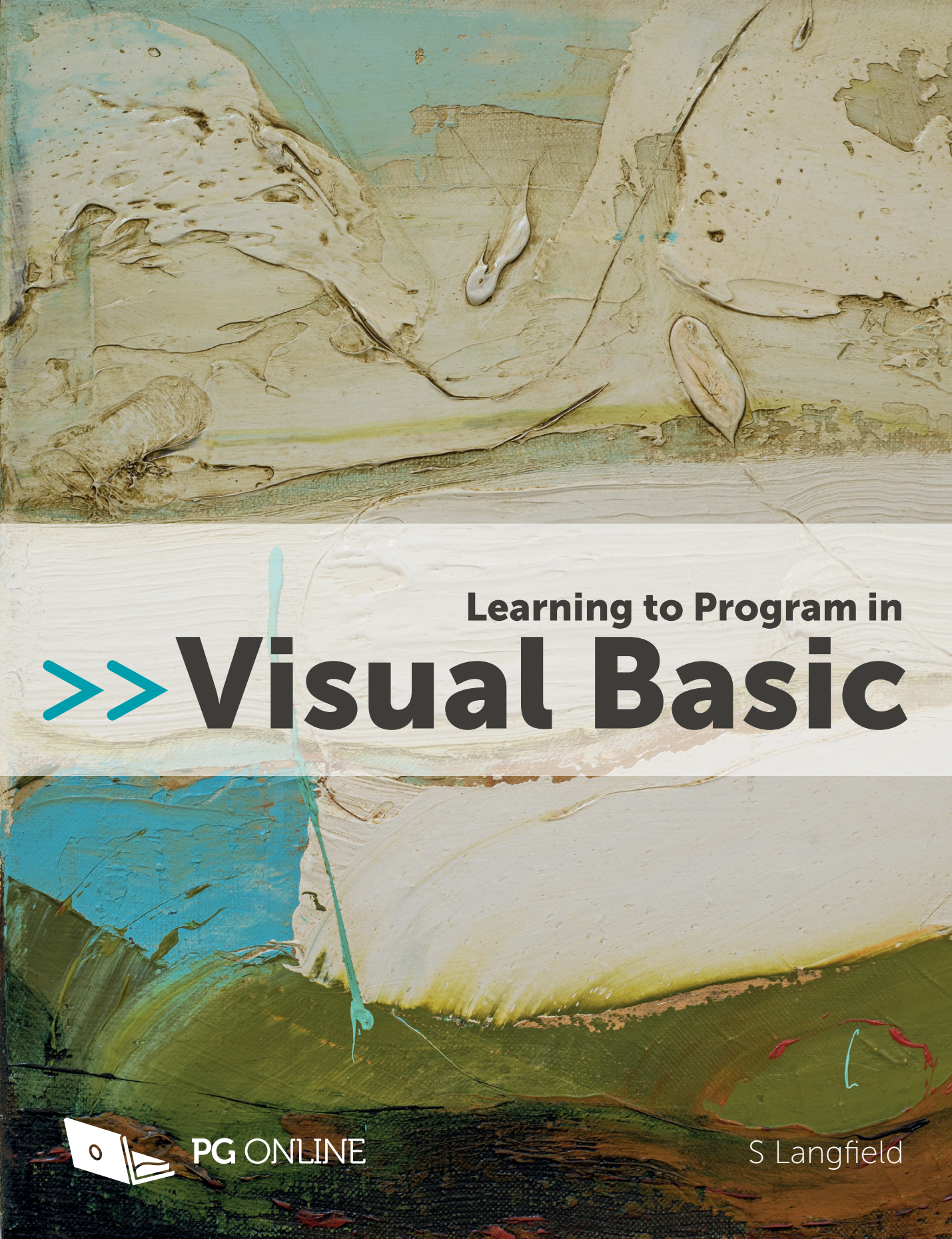


Learning to Program in

>>Visual Basic

PG ONLINE

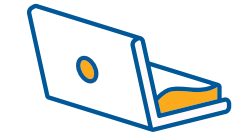
S Langfield

Learning to Program in

Visual Basic .NET >>

S Langfield

Published by
PG Online Limited
The Old Coach House
35 Main Road
Tolpuddle
Dorset
DT2 7EW
United Kingdom
sales@pgonline.co.uk
www.pgonline.co.uk



PG ONLINE

Graphics: PG Online Ltd
Design and artwork: PG Online Ltd
First edition 2019
A catalogue entry for this book is available from the British Library
ISBN: 978-1-910523-18-6

Printed on FSC certified paper
Printed by Bell and Bain Ltd, Glasgow, UK.

Preface

Programming is fun! Trial and error is to be encouraged and you should type all of the examples and try all the exercises to get used to entering and debugging programs and to see how the programs run.

This book is intended for individuals and students who may have done some programming in other languages, but are not familiar with Visual Basic .NET. It is intended that users of the book should work through the book sequentially, starting at Chapter 1. However, it will be a useful reference book for students on a programming course or anyone working on a programming project.

It teaches basic syntax and programming techniques and introduces a number of useful features such as:

- **Developing graphical user interfaces** (GUIs) with the visual designer in Visual Studio.
- **SQLite**, which enables the creation and processing of a database from within a Visual Basic .NET program. This provides an alternative to writing to a text file when data needs to be stored and retrieved.
- **The Visual Studio debugger**, which can be used to help find elusive logic errors.

Questions and exercises are included throughout every chapter. Over 120 VB programs for all the examples and exercises given in the book may be downloaded from **www.pgonline.co.uk**. We strongly advise you to write your own code and check your solutions against the sample programs provided.

This book is a companion volume to the book Learning to Program in Python by P.M. Heathcote, and uses the same format. Questions and exercises from that text have been used throughout this book, with answers and programs rewritten in Visual Basic.

Enjoy – the sky's the limit!

Downloading Visual Basic .NET

VB.NET is a high-level programming language, implemented on the .NET Framework. Microsoft® launched VB.NET in 2002 as the successor to its original Visual Basic language. Microsoft's integrated development environment (IDE) for developing programs in VB.NET is called Visual Studio. Visual Studio Express and Visual Studio Community are freeware. VB.NET is available to be downloaded free from https://visualstudio.microsoft.com/vs/express/. The programs have been written and tested in Visual Studio Express 2019. Many schools and individuals may prefer to use alternative development environments and the book is equally applicable to these.

Contents

Chapter 1

Input, output and assignment

Objectives

- Write a simple console application
- Use string, numeric and Boolean data types and operators
- Learn the rules and guidelines for declaring and naming variables
- Use input and output statements

Programming in VB .NET

Visual Basic (VB) is a popular programming language. A VB program is simply a series of instructions written according to the rules or **syntax** of the language, usually designed to perform some task or come up with a solution to a problem. You write the instructions, and then the computer translates these instructions into binary machine code which the computer can execute. It will do this using a translator program, called a **compiler**.

VB.NET is the latest version of the Visual Basic programming language. It comes with an **integrated development environment** (IDE) which enables you to enter your program, save it, edit it, translate it to machine code and run it once it is free of syntax errors. If you have written a statement incorrectly, it will be reported by the IDE or the compiler as a syntax error, and you can correct it and try again.

You can write simple programs as a console application or you can write Windows® based applications using Windows forms (see Chapters 12 and 13).

Augmented assignment operators

These operators provide a shortcut to writing assignment statements.

Operator	Example	Equivalent to
+=	`score += 1`	`score = score + 1`
-=	`score -= losses`	`score = score - losses`
*=	`score *= 2`	`score = score * 2`
/=	`score /= total`	`score = score / total`
\=	`score \= 7`	`score = score \ 7`

Table 1.4: Augmented assignment operators

Q4 Write statements using augmented assignment operators to do the following:

(a) Add `1` to `counter`.

(b) Double a variable called `housePrice`.

(c) Subtract a variable called `penalty` from a variable called `hits`.

(d) Divide `totalCostOfMeal` by `3`.

The `Write` and `WriteLine` statements

You have already seen the `Console.Write` and `Console.WriteLine` statements used to display text on the screen.

Using an ampersand (&) as a concatenation operator

You can use **&** to separate strings or variables:

```
Dim length As Integer = 15
Console.WriteLine("The length is " & length & " metres")
```

produces the output:

```
The length is 15 metres
```

Using `&` has the advantage that you do not have to worry about the variable type. You can mix integers, real numbers and strings in the `Write` statement.

Using + as a concatenation operator

Instead of using `&`, you can use a `+` operator to concatenate strings in the output. However, you cannot join a string to an integer or real number, so you first have to convert any numbers to strings.

Q1 What will happen if you write

```
Dim name = {"Mark", "Juan", "Ali", "Cathy",
"Sylvia", "Noah"}
For index = 0 To 6
    Console.WriteLine(name(index))
Next
```

Operations on arrays

Some array methods are shown in the table below. Assume `a = {45,13,19,13,8}`.

Array operation	Description	Example	array contents	Return value
Count()	Counts the elements in the array	`a.Count()`	{45, 13, 19, 13, 8}	5
Length()	Return the number of elements	`a.Length()`	{45, 13, 19, 13, 8}	5
Contains(Item)	Returns True if Item exists in array, False otherwise	`a.Contains (13)`	{45, 13, 19, 13, 8}	True
Max()	Returns the largest value	`a.Max()`	{45, 13, 19, 13, 8}	45
Min()	Returns the smallest value	`a.Min()`	{45, 13, 19, 13, 8}	8
Sum()	Returns the sum of all elements of a numeric array	`a.Sum()`	{45, 13, 19, 13, 8}	98
Average()	Returns the average of all elements of a numeric array	`a.Average()`	{45, 13, 19, 13, 8}	19.6

Table 5.1 Array operations

Example 2

Determine whether the number `100` is in the array `numbers = {56,78,100,45,88,71}`, and if so, print its index.

```
'Program name: Ch 5 Example 2 Array of numbers
Dim numbers = {56, 78, 100, 45, 88, 71}
Dim index = Array.IndexOf(numbers, 100)

If index = -1 Then
    Console.WriteLine("100 is not in the array")
Else
    Console.WriteLine("100 is at index number: " & index)
End If
```

An array is a static data structure, that means it is fixed in size. The elements of an array must all be of the same data type.

produces the output:

```
Key:  Wesley      Value: 5
Key:  Jo          Value: 9
Key:  Betty       Value: 6
Key:  Robina      Value: 5
```

To look up the mark obtained by a particular student (the key), write the name of the dictionary followed by the key in brackets. You cannot index a dictionary in the same way as an array, using an index number – an item can only be accessed through its key.

The statement

```
Console.WriteLine("Betty: " & studentMarks("Betty"))
```

produces the output:

```
Betty: 6
```

Q1 Write a statement to print the mark obtained by Robina.

The table below shows some of the most useful built-in dictionary methods.

Method	Description
ContainsKey(key)	Finds if a key is present in the dictionary
Add(key, value)	Adds a key to the dictionary
Keys	Returns all the keys in the dictionary
Remove(key)	Removes an item from the dictionary

Table 7.1: Useful dictionary methods

Example 1a: Looking up a value

If you try to print the mark for a student whose name is not in the dictionary, it will return an error. To avoid this, you should test whether the key is in the dictionary before trying to access it.

```
Dim studentMarks As New Dictionary(Of String, Integer) From
  {{"Wesley", 5}, {"Jo", 9}, {"Betty", 6}, {"Robina", 5}}
Dim name As String
Console.Write("Enter a student name to look up: ")
name = Console.ReadLine()
If studentMarks.ContainsKey(name) Then
    Console.WriteLine("Mark: " & studentMarks(name))
Else
    Console.WriteLine("Name not found")
End If
```

Example 6

Write a program that uses StreamWriter to create a new file called **temperatures.txt**, or append records to the file if one already exists.

```
Imports System.IO
' Program name: Ch 9 Example 6 writing to a file using
StreamWriter
Module Module1
    Sub Main()
        Dim tempsFile As New StreamWriter("temperatures.txt",
        True)
        Dim city, localTime As String
        Dim temperature As Integer
        Console.WriteLine("Writes data to temperatures.txt")
        Console.WriteLine("If file does not exist, it will
                           be created")
        Console.Write("Enter city name, xxx to end: ")
        city = Console.ReadLine()
        Do While Not city = "xxx"
            Console.Write("Enter temperature: ")
            temperature = Console.ReadLine()
            Console.Write("Enter local time: ")
            localTime = Console.ReadLine()
            tempsFile.WriteLine(city & "," & temperature &
                                "," & localTime)
            Console.Write("Enter city name: ")
            city = Console.ReadLine()
        Loop
        tempsFile.Close()
        Console.Write("Press Enter to exit ")
        Console.ReadLine()
    End Sub
End Module
```

Q1 Write a program to read and print all the records in temperatures.txt.

Example 7

Read the data in the file **temperatures.txt**, convert all the Centigrade temperatures to Fahrenheit and print out both the Centigrade and Fahrenheit temperatures.

```
Imports System.IO
' Program name: Ch 9 Example 7 process temperatures file
Module Module1
    Sub Main()
        Dim tempsFile = 1
        Dim city, localTime As String
        Dim temperatureC As Integer
        Dim temperatureF As Double
```

Controls

Controls are GUI objects, such as buttons and text entry fields, that are used to interface with the program. They can also be used to display information to the user in the form of a label or a graphic. When adding a control to the form designer, it is a good idea to give it a sensible name, so that your program code will be easier to maintain. The convention is to use a prefix with each control identifier that represents the control's type. For example, the identifier for a confirm button might be `btnConfirm`.

Here is a list of prefixes for some common components along with an example:

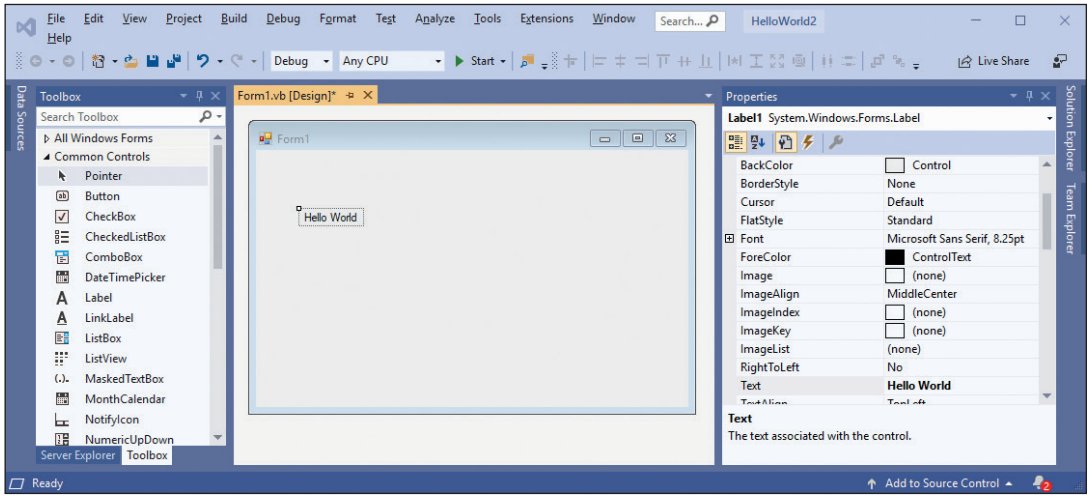
Components	Prefix	Example
Button	`btn`	`btnConfirm`
Checkbox	`chk`	`chkSunday`
Form	`frm`	`frmConfirmation`
Label	`lbl`	`lblInstructions`
Textbox	`txt`	`txtFirstName`
ComboBox	`cbo`	`cboTasks`
GroupBox	`grp`	`grpDeliveryMethod`

The "Hello World" program

Example 1

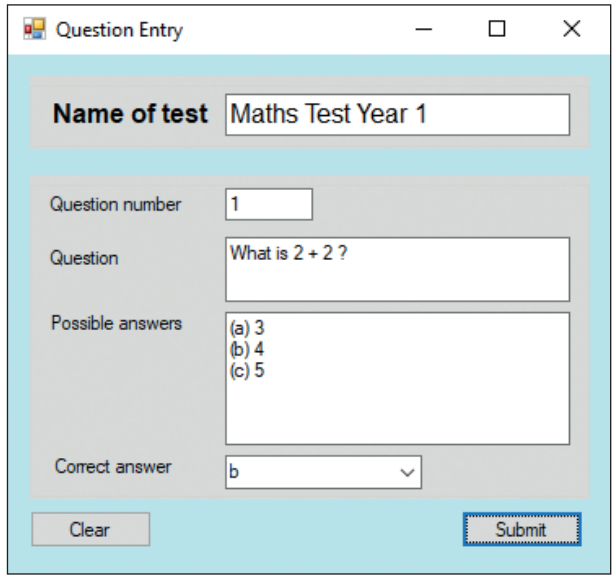
Select the **Label** component from the **toolbox** and drag it onto the form.

In the **Properties** window change the **Text** property to "`Hello World`".



Sample application 3

This application allows a user (for example, a teacher) to create a multiple-choice test consisting of several questions which could be saved in a text file or database. The input window will look like this:



Create a new **Windows Forms App** project and add the following components to the form. The values for key **properties** are shown below.

Component	Property	Value
Form1	Text	Question entry
	BackColor	192, 255, 255
GroupBox1	Text	""
	BackColor	224, 224, 224
Label1	Text	Name of test
	Font	Arial, 12pt, Bold
TextBox1	Name	txtTestName
GroupBox2	Text	""
	BackColor	224, 224, 224
Label2	Text	Question number
Label3	Text	Question
Label 4	Text	Possible answers
Label5	Text	Correct answer:
TextBox2	Name	txtQuestionNumber

Index

Learning to Program in Visual Basic >>

This book is intended for individuals and students learning to program. You may already have done some programming in other languages, but not be familiar with Visual Basic. Novice programmers should work through the book sequentially, starting at Chapter 1. It will also be a useful reference book for students on a programming course or anyone working on a programming project.

It teaches basic syntax and programming techniques, and introduces databases, SQL and SQLite, and the development of a graphical user interface for a Windows application.

Questions and exercises are included in every chapter. Answers to these as well as over 120 VB programs for all the examples and exercises given in the book may be downloaded from **www.pgonline.co.uk**. These programs enable users of the book to try out the in-text examples and check possible solutions to the exercises.

Cover picture:
'Fosse No1'
Oil on linen,
30x30cm © Barbara Burns 2015
www.slaneart.com

ISBN 9781910523186

PG ONLINE