Table of Contents

Introduction

This book covers the basics of Visual Basic.net.

Introduction to Console Applications

Console applications are programs that interact with users using a text only interface. The most used console application is perhaps the "Command Prompt" in Microsoft Windows. It can be opened by finding "Command Prompt" in the Start Menu, or by running the "**cmd**" command.

Comparison with GUI applications

Most computer users use only **GUI (graphical user interface) applications**, with graphical elements like buttons, text boxes, pull-down menus, etc. While GUI applications are more satisfying to users, they are not easy to create. Instead, console applications are straight forward even for beginners.

Therefore, you are learning to create console applications first. When you acquire enough experience, you can learn to create GUI applications.

Your first VB.NET program

Now you will create your first VB.NET program.

Creating a new project

This book assumes that you're using Visual Studio 2015 Community. The user interface may be a bit different in other versions of Visual Studio, but the interface should be similar.

After you start Visual Studio, select "New Project" in the File menu. Select "Visual Basic" in the left pane, and then "Console Application" in the right pane. Give the project a name (if not "ConsoleApplication1"), and then press okay. A project with a file called `module1.vb` will be created.

Adding some code

Now you should add your own code between `Sub Main()` and `End Sub`. Your file should look like this:

> hello_world.vb

```
Module Module1

    Sub Main()
        Console.WriteLine("Hello world!")
        Console.ReadLine()
    End Sub

End Module
```

Running the program

Press the key `F5` to run the program. After seeing the result, press the key `Enter` to close the program.

Using the code view

For those familiar with word processors, the code editor should be very easy to use. However, this is not the case for an average student. So it will be advantageous to try the following:

- *Remember the shortcut key `Ctrl+Z` for undo. This is the FIRST thing to do when something is wrong.*
- Use the keys `Insert`, `Delete`, `Backspace` and `Enter`. Do this
  - at the beginning of a line, and
  - in the middle of a line.
- Select text by holding `Shift` while using the arrow keys.

- Use the shortcut keys `Ctrl+C` (copy), `Ctrl+X` (cut) and `Ctrl+V` (paste).
- Type when there is an active selection. See how the text is overwritten.
- Advanced: While holding `Ctrl`, press the left and right arrow keys to move the cursor. You may use this in combination with the `Shift` key.

The Console object

In VB.NET the console is accessed by the `Console` object. It has the following methods:

Method	Description	Example
WriteLine	Outputs a string to the console, and moves the cursor to the next line	`Console.WriteLine("str")` `Console.WriteLine()`
Write	Outputs a string to the console, but do not move the cursor to the next line.	`Console.Write("str")`
Clear	Clear the content of the console. Also moves the cursor to the top-left corner.	`Console.Clear()`
ReadLine	Reads a line from the console (press `Enter` to finish input). In this book, ReadLine method is also used to prevent the console from closing after running the program.	`Dim s As String` `s = Console.ReadLine()`

Console.WriteLine and Console.Write

`Console.WriteLine` and `Console.Write` are used to output text to the console. These methods has only one difference: `Console.WriteLine` moves the cursor to the next line, while `Console.Write` does not.

Calling `Console.WriteLine` without any argument simply moves the cursor to the next line.

The example below shows the differences of Console.WriteLine and Console.Write.

> console_writeline_and_write.vb

```
Module Module1
    Sub Main()
        Console.WriteLine("Console.WriteLine results")
        Console.WriteLine("==========================")
        Console.WriteLine("Item 1")
        Console.WriteLine(2.345)
        Console.WriteLine("Item " & 3)
        Console.WriteLine("Item {0}", 4)
        Console.WriteLine()
        Console.WriteLine()
        Console.WriteLine("Console.Write results")
        Console.WriteLine("=====================")

        Console.Write("Item 1")
        Console.Write(2.345)
        Console.Write("Item " & 3)
        Console.Write("Item {0}", 4)

        Console.ReadLine()
    End Sub
End Module
```

See the output below to check your understanding.

```
Console.WriteLine results
==========================
Item 1
2.345
Item 3
Item 4


Console.Write results
=====================
Item 12.345Item 3Item 4
```

Substitutions in the output

If `{0}`, `{1}`, `{2}`, ... are found in the string, it is replaced by following arguments (i.e. the things inside `Console.WriteLine` and the like, separated by commas).

Since the output string is the first argument, `{0}` is replace by the content of the second argument, `{1}` by the third argument, etc.

See the following example:

> console_writeline_substitutions.vb

```
Module Module1
    Sub Main()
        // {0} is "David", {1} is "Peter".
        Console.WriteLine("{0} and {1} are good friends.",
                          "David", "Peter")
        // {1} is used multiple times.
        Console.WriteLine("{1} * {1} = {0}", 9 * 9, 9)
        Console.ReadLine()
    End Sub
End Module
```

Class Work

Write a program that produces the given output. Use `Console.WriteLine` only.

Output	Visual Basic Program

`This is my first console program.`
 	
`It is easy!`
	`Sub Main()`
	
	`End Sub`

Console.Clear

`Console.Clear` is used to clear the content of the console. After the console is cleared, the cursor goes to the top-left corner.

Console.ReadLine

The Console.ReadLine method reads a line of characters from the console and returns a string. To add a prompt, use Console.Write immediately before Console.ReadLine.

A typical use of Console.ReadLine is as follows:

```
Console.Write("Enter the number n: ")
Dim n As Integer = Console.ReadLine()
```

Again, to convert the string into a number, use the Val function. Example 1.2.2 Console.ReadLine with prompt: Find the n-th triangle number. Module Module1 Sub Main() Console.Write("Enter the number n: ") Dim n As Integer = Val(Console.ReadLine()) Console.WriteLine("The " & n & "th triangle number is " & n * (n + 1) \ 2) Console.ReadLine() End Sub End Module In the notes, Console.ReadLine is also used at the end of Sub Main. This prevents the console from closing when the program ends.