VB.NET Code Bank

for KS4 Computer Science

POD 7701

BW11.7701

computerscience@zigzageducation.co.uk
zigzageducation.co.uk

Contents

Teacher's Introduction

This resource has been written to provide students with explanations and examples of the core programming techniques available in the Visual Basic .NET programming language.

The range and complexity of the techniques and examples covered in this resource make it ideal for KS4 level (it has been produced with GCSE Computer Science specifications in mind) – however it could be used at any key stage where students are learning to program. For example, by familiarising students with the resource during KS3 lessons, they will know how to make use of it at GCSE.

Students can then refer to the syntax, and adapt code for use in their own programs.

Important: if you are intending to use this resource to support students while working on their non-exam assessments (NEA), it is your responsibility to ensure that the support you provide students with is appropriate, including meeting any guidelines set out by your exam board.

The techniques covered have been broken into 39 different topics, each consisting of the following:

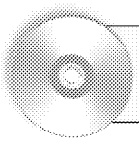
1. *Description of the code* – detailing the purpose of the code, and the valid syntax structure needed.
2. *Code in context* – a series of short, generic code snippets showing examples of each technique in use. Each one is summarised in plain English, with comments throughout the code to explain how it works.

Each topic is provided as one or more separate A4 pages, making it easy for you to select the ones you want to hand out to students. A Word version is also provided on disk, allowing you to edit and print the worksheets – including in colour should you want to.

In addition to the paper formats, the code snippets are also provided electronically in the following ways:

1. As 152 individual *TXT* files, from which students can copy and paste the code into an integrated development environment of their choice and adapt for their own programs.
2. A *HTML* interface includes all of the snippets, with a side menu enabling easy navigation between them.

This resource may be used on your school network by copying the files from the CD to a location which is accessible to students.



The CD contains three folders: one containing the code snippets in **TXT** format, one with the code snippets in **HTML** format, and one containing a **DOCX** version (for editing/printing from MS Word).

Output: Conso

Description of Code

Output allows the user to print data to the screen for the user to read.

The code:

Console.WriteLine(<dat o utput>)

outputs the data inside the brackets to the screen. The data can be a string, e such as the

Multiple values for outputting need to have & symbols between them (see exa

Console.WriteLine(<data to output> & <data to ou

Code in Context

1. The text '*Hello World*' is outputted.

```
Console.WriteLine("Hello World") 'output "Hello World"
```

2. A variable is used to store a value; which is then ou ed.

```
'declare a new variable called
Dim theNumber As Integer

'store the numb the variable theNumber
theNum =

'outp value in theNumber
Console.WriteLine(theNumber)
```

3. A variable is used to store a value; which is then outputted along with so

```
'declare a new variable called theNumber
Dim theNumber As Integer

'store the value 12 in the variable theNumber
theNumber = 12

'output the value of the variable theNumber and the text "
Console.WriteLine(theNumber & " is a number")
```

4. A variable is used to store a value which is then outputted along with so before and after it.

```
'declare a variable called theNumber as an Integer
Dim t e As Integer

'store the number 12 in the variable theNumber
theNumber = 12

'output the value of the variable theNumber surrounded by t
Console.WriteLine("the number " & theNumber & " is my favou
```

Output: MsgBo

Description of Code

Output allows the user to print data to the screen for the user to read.

The code:

MsgBox(<data to output>)

outputs the data inside the brackets to the screen in a new dialog box. The da
e.g. "Hello World" or a variable such as theNumber.

Multiple values outputting need to have **&** symbols between them; for exa

MsgBox(<data to output> & <data to output>)

Code in Context

*Each example uses a form with a button named **enterButton**. The code is run wh*

1. The program outputs "Hello World" in a message box.

```
'the procedure call for the button being clicked
Private Sub enterButton_Click(sender As Object, e As EventArgs) Ha
    MsgBox("Hello World") 'output "Hello World" in a message box
End Sub
```

2. The program stores the number 12 in a variable and then outputs the val

```
'the procedure call for the button being clicked
Private Sub enterBut...k(sender As Object, e As EventArgs) Ha
    Dim theNum... Integer 'declare a new variable called theNu
    th...r 12 'store the number 12 in the variable theNumber
    Msg...eNumber) 'output the contents of the variable theNum
End Sub
```

3. The program stores the number 12 in a variable, and then outputs the val
the text " is a number".

```
'the procedure call for the button being clicked
Private Sub enterButton_Click(sender As Object, e As EventArgs) Ha
    Dim theNumber As Integer 'declare a new variable with the nam
    theNumber = 12 'store the number 12 in the variable theNumber

    'in a msgbox, output the contents of theNumber, followed by a
    MsgBox(theNumber & " is a number")
End Sub
```

4. The program stores the number 12 in a variable then outputs the text "the
in the variable, and then the text "...vourite number".

```
'the procedure ca...r button being clicked
Private ...e...n_Click(sender As Object, e As EventArgs) Ha
    Di...um...r As Integer 'declare a new variable with the nam

    'st... the number 12 in the variable theNumber
    theNumber = 12

    'in a msgbox, output the value of theNumber surrounded by two
    MsgBox("the number " & theNumber & " is my favourite number")
End Sub
```

Output: Objec

Description of Code

Output allows the user to print data to the screen for the user to read.

In addition to using MsgBox, you can also output to labels and other objects.

The code:

<object name>.Text = <data to output>

outputs <data to output> on the right-hand side of the = to the object <object name>.
The data can be a string, e.g. "Hello World", or a variable such as theNumber.

Multiple values for outputting need to have & symbols between them in the same line.

Code in Context

Each example uses a form with a command button and a label.

1. The program outputs the text "Hello World" to a label.

```
'the procedure call for when the button is clicked
Private Sub enterButton_Click(sender As Object, e As EventArgs) Ha
    testLabel.Text = "Hello World" 'write the text "Hello World"
End Sub
```

2. The program stores the number 12 in a variable, and then writes the conte

```
'the procedure call for when the button is clicked
Private Sub enterButton_Click(sender As Object, e As EventArgs) Ha
    Dim theNumber As Integer 'declares a new variable with the na
    theNumber = 12 'stores the number 12 in the variable theNumbe
    testLabel.Text = theNumber 'outputs the contents of theNumber
End Sub
```

3. The program stores the number 12 in a variable, and then writes the conte
" is a number" to a label.

```
'the procedure call for when the button is clicked
Private Sub enterButton_Click(sender As Object, e As EventArgs) Ha
    Dim theNumber As Integer 'declares a new variable with the na
    theNumber = 12 'stores the number 12 in the variable theNumbe

    'outputs the contents of theNumber, followed by a string, to
    testLabel.Text = theNumber & " is a number"
End Sub
```

4. The program stores the number 12 in a variable, and then writes the text "
theNumber and the text "is my favourite number" to a label.

```
'the procedure call for when the button is clicked
Private Sub enterButton_Click(sender As Object, e As EventArgs) Ha
    Dim theNumber As Integer 'declares a new variable named theNu
    theNumber = 12 'writes the number 12 to the variable theNumbe

    'outputs the value of the theNumber, surrounded by two string
    testLabel.Text = "the number " & theNumber & " is my favourit
End Sub
```

Input: Consol

Description of Code

Input allows the user to enter some data that can then be used in the program.

The code:

Console.ReadLine

waits for the user to enter some text and press the return/enter key.

Code in Context

1. The program reads the data from the console.

```
'the program waits for the user to enter something and pres
'it does nothing with this data.
Console.ReadLine()

'outputs the text
Console.WriteLine("Read data")
```

2. The program reads the data from the console and stores it in the variable t

```
'declares a new variable named theNumber
Dim theNumber As String

'stores the data the user enters in the variable theNumber
theNumber = Console.ReadLine

'outputs the value in theNumber
Console.WriteLine(theNumber)
```

3. The program reads the data from the console and outputs it to the console

```
'reads the text the user enters and outputs this to the scr
Console.WriteLine(Console.ReadLine)
```

Input: Text Bo

Description of Code

Input allows the user to enter some data that can then be used in the program.

VB.NET has different visual objects that can be used to get the user to enter te

Text Box

To access what the user has put in a text box, use:

```
<textbo me>.Text
```

Code in Context

*Both examples use a text box, **TextBox1**, and a command button with the name B*

1. The program reads the data in the text box TextBox1 and stores it in the v

```
'the procedure call for when the button is clicked
Private Sub Button1_Click(sender As Object, e As EventArgs)

    'declares a variable called theValue
    Dim theValue As String

    'stores the data in the text bo  h  o TextBox1 in the
    theValue = TextBox1.Text

    'outputs the va  ir  heValue in a message box
    msgbox(th      )
End Su
```

2. The program reads the data in the text box Textbox1 and outputs it in a m

```
'the procedure call for when the button is clicked
Private Sub Button1_Click(sender As Object, e As EventArgs)

    'outputs the data in the text box TextBox1
    MsgBox(TextBox1.Text)
End Sub
```

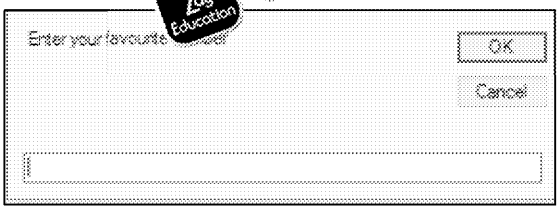
Input: Input B

Description of Code

Input allows the user to enter some data that can then be used in the program.
VB.NET has different visual objects that can be used to get the user to enter te

Input Box

A window appears on screen that the user can type data into.
The box will have either OK, or cancel; for example:



The code for an input box is:

```
InputBox("<text to output>")
```

Code in Context

*In each example there is a command button with the name **Button1** that is clicke*

1. The program outputs the text "Enter some information" in an input box an input box.

```
'the p      e call for when the button is clicked
Private      Button1_Click(sender As Object, e As EventArgs) Handl

    'displays an InputBox with the text "Enter some information"
    InputBox("Enter some information") '
    Msgbox("Read data") 'outputs the text
End Sub
```

2. The program displays an input box and stores the text the user inputs in t

```
'the procedure call for when the button is clicked
Private Sub Button1_Click(sender As Object, e As EventArgs) Handl
    Dim theNumber As Integer 'declares a variable called theNumbe

    'displays an InputBox with the text "Enter your favourite num
    'data entered in the variable theNumber
    theNumber = InputBox("Enter your favo      ber")
    Msgbox(theNumber) 'outputs the v      heNumber
End Sub
```

3. The program dis   input box, lets the user enter text, and then outp

```
'the p      e call for when the button is clicked
Private      Button1_Click(sender As Object, e As EventArgs) Handl

    'displays a message box saying "Enter your favourite number"
    'entered by the user
    MsgBox(InputBox("Enter your favourite number"))
End Sub
```

Variables

Description of Code

A variable is a space in memory that stores a piece of data that can change. You give the location a name so it can be easily accessed. You can put data in the memory and get data out of it.

You need to know what type of data you will be storing. Once you have decided you cannot change it to a different type of data later (unless you change the code).

To use a variable you must declare it first:

```
Dim <variable name> as <data type>
```

Data types

Name	Description	
Integer	Whole numbers	*0, 123*
Boolean	True or False	*True, F*
String	Characters, including symbols and numbers that do not need to be used in mathematical calculations	*"Hello*
Single	Decimal numbers	*0.0, 23*
Char	A single character, including a symbol or number that does not need to be used in mathematical calculations	*"B", "#"*
Date		*"22/03*

Putting data in a variable

```
<variable name> = <data or expression>
```

The = can be read as 'becomes', so the variable on the left becomes the data or

For example, in this code the variable myVariable becomes the number 123.

```
myVariable = 123
```

Getting data from a variable

To access the data in a variable, use its name.

For example, to output the data in a variable to the console:

```
Console.WriteLine(<variable name>)
```

Code in Context

1. A variable called **myNumber** has the number stored in it.

```
Dim myNumber As Integer 'declare myNumber as a whole n
myNumber = 123 'myNumber becomes 123
Console.WriteLine(myNumber) 'output the value in myNum
```

2. A variable called **favouriteFilm** as a string has "The Matrix" stored in it.

```
Dim favouriteFilm As String 'declare favouriteFilm as
favouriteFilm = "The Matrix" 'favouriteFilm becomes "T
Console.WriteLine(favouriteFilm) 'output the value in
```

3. A variable called **continueFlag** as a Boolean has True stored in it.

```
Dim continueFlag As Boolean 'declare continueFlag as a
continueFlag = True 'continueFlag becomes True
Console.WriteLine(continueFlag) 'output the value in c
```

4. A variable called **myNumber** is given the value of 2.5, which is then outpu

```
Dim myNumber As Single 'declare myNumber as a Single
myNumber = 2.5 'myNumber becomes 2.5
Console.Write(myNumber) 'output the value in myNumber
```

5. A variable called **cityName** is given the value of "London", which is then o

```
Dim cityName As String 'declare cityName as a String
cityName = "London" 'cityName becomes "London"
Console.Write(cityName) 'output the value in cityName
```

Constants

Description of Code

A constant is a space in memory that stores a piece of data that cannot change. We give the memory location a name so it can be easily accessed. You have to put data in when you declare the constant; you cannot change this later. You can get the data out of it.

You need to know what type of data you will be storing – this is the same as with variables.

To use a constant you must declare it first:

```
Const <constant name> as <data type> = <value>
```

Getting data from a constant

To access the data in a constant, use its name.

For example, to output the content of a constant to the console:

```
Console.Writeline(<constant name>)
```

Code in Context

1. Declare a constant called **myNumber** as an integer and store the number 123 in it.

```
'declare myNumber as integer
Const myNumber As Integer = 123

'output the value in myNumber
Console.WriteLine(myNumber)
```

2. Declare a constant called **vatRate** as a single and store 0.2 in it.

```
'declare vatRate as the single value 0.2
Const vatRate As Single = 0.2

'output the value in vatRate
Console.WriteLine(vatRate)
```

3. Declare a constant called **numTimes** as an integer and store 10 in it. Output the value in numTimes.

```
'numTimes is a constant with the Integer value 10
Const numTimes As Integer = 10

'output the value in numTimes
Console.Write(numTimes)
```

Local and Global Va

Description of Code

Variables and constants can be declared in different places within a program.

If they are declared within a subroutine then they are local.

This means they are created when the subroutine is called; they exist within t and when the subroutine exits they disappear. They cannot be accessed outsi they are declared in.

If they are declared at the top of the program, outside any subroutines, then th are created as soon as the program starts running, they exist throughout the p stops they disappear. They can be accessed by any part of the program.

Global variables are inefficient because they take up memory even when they a can be used to send data between subroutines, but it is simpler to program wi

Think carefully about whether you need a global variable or whether a local w

Code in Context

1. Read values into two global variables, output the variables in Main() and

```
Dim num1 As Integer 'declare num1 as a global integer
Dim num2 As Integer 'declare num2 as a global integer

Sub Main()
    num1 = Console.ReadLine() 'read the user input and store in t
    num2 = Console.ReadLine() 'read the user input and store in t

    'output the values in num1 and num2
    Console.WriteLine(num1 & " and " & num2 & " exist in the Main

    nextProcedure()  'call the procedure nextProcedure
    Console.ReadLine()
End Sub

Sub nextProcedure() 'declare the procedure nextProcedure, without

    'output the values in num1 and num2
    Console.WriteLine(num1 & " and " & num2 & " exist in nextProc
End Sub
```

2. Read values into two local variables to Main(), output them in Main() but

```
Sub Main()
    Dim num1 As Integer 'declare num1 as a local integer to the M
    Dim num2 As Integer 'declare num2 as a local integer to the M

    num1 = Console.ReadLine() 'read the user input and store in t
    num2 = Console.ReadLine() 'read the user input and store in t

    'output the values in num1 and num2
    Console.WriteLine(num1 & " and " & num2 & " exist in the Main

    nextProcedure()  'call the procedure nextProcedure
    Console.ReadLine()
End Sub

Sub nextProcedure() 'declare the procedure nextProcedure, without

    'output the string
    Console.WriteLine("num1 and num2 do not exist in this procedu
End Sub
```

3. Reads values into two global variables. Outputs the values, changes the v
and then output the new values.

```
Dim num1 As Integer 'declare num1 as a global integer
Dim num2 As Integer 'declare num2 as a global integer

Sub Main()
    num1 = Console.ReadLine() 'read the user input and store in t
    num2 = Console.ReadLine() 'read the user input and store in t

    'output the values in num1 and num2
    Console.WriteLine(num1 & " and " & num2 & " exist in the Main

    nextProcedure() 'call the procedure nextProcedure

    'output the new values in num1 and num2
    Console.WriteLine("The new values are " & num1 & " " & num2)
    Console.ReadLine()
End Sub

Sub nextProcedure() 'declare the procedure nextProcedure, without
    num1 = 10 'store 10 in num1
    num2 = 20 'store 20 in num2

    'output the string
    Console.WriteLine("num1 and num2 have been changed in the pro
End Sub
```

4. Read values into two local variables in the Main() procedure. Outputs the
Declares local variables to nextProcedure with the same identifiers, store
outputs them in nextProcedure. Re-outputs the original values in Main().

```
Sub Main()
    Dim num1 As Integer 'declare num1 as a local integer to the M
    Dim num2 As Integer 'declare num2 as a local integer to the M

    num1 = Console.ReadLine() 'read the user input and store in t
    num2 = Console.ReadLine() 'read the user input and store in t

    'output the values in num1 and num2
    Console.WriteLine(num1 & " and " & num2 & " exist in the Main

    nextProcedure() 'call the procedure nextProcedure

    'output the values in num1 and num2
    Console.WriteLine(num1 & " and " & num2 & " have not been cha

    Console.ReadLine()
End Sub

Sub nextProcedure() 'declare the procedure nextProcedure, without
    Dim num1 As Integer = 10 'declare num1 local to nextProcedure
    Dim num2 As Integer = 20 'declare num2 local to nextProcedure

    'output the string
    Console.WriteLine(num1 & " and " & num2 & " have been redecla
End Sub
```

Casting

Description of Code

Some data can be converted to a different data type.

For example, the string "123" could be converted to the integer 123.

The code:

CType(<variable name>, Integer)

converts the given variable to an integer.

Instead of Integer we can also use:

- **String** to convert to a string
- **Single** to convert to a decimal number

Code in Context

1. The program reads a string from the user, converts it to an integer, adds 10 to a string to output.

```
Dim userInput As String
Dim userNumber As Integer
Console.WriteLine("Enter a number") 'output the text

'read the string and store it in the variable userInput
userInput = Console.ReadLine()

'convert the value in userInput to integer and store the re
userNumber = CType(userInput, Integer)
userNumber = userNumber + 10 'add 10 to the value in userNu
Console.WriteLine(userNumber) 'output the value in userNumb
```

2. The program reads two numbers from the user, converts each to a decimal variables. These decimal numbers are added together. The total is conve

```
Dim cost1Str, cost2Str As String
Dim cost1, cost2, total As Single

Console.WriteLine("Enter cost 1") 'output the text

'read the value as a string, store it in the variable cost1
cost1Str = Console.ReadLine
Console.WriteLine("Enter cost 2") 'output the text

'read the value as a string, store it in the variable cost2
cost2Str = Console.ReadLine

'convert cost1Str and cost2Str to decimal, store the result
cost1 = CType(cost1Str, Single)
cost2 = CType(cost2Str, Single)

'add together the values in cost1 and cost 2, store the res
total = cost1 + cost2

'output the value in the variable total
Console.WriteLine(total)
```

3. Read in a number as a string, and store it in **stringValue**. Convert it to a de
floatValue. Convert the value in **floatValue** to a whole number and store i
Output all three as strings.

```
Dim stringValue As String
Dim singleValue As Single
Dim integerValue As Integer

Console.WriteLine("Enter a number") 'output the text

'read the input as a string, store it in the variable strin
string... Console.ReadLine()

'conve... value in stringValue to a decimal and store it
singleValue = CType(stringValue, Single)

'convert the value in singleValue to a decimal and store it
integerValue = CType(singleValue, Integer)

'output the text and the value in stringValue
Console.WriteLine("The string value is " & stringValue)

'output the text and the value in singleValue
Console.WriteLine("The single value is " & singleValue)

'output the text and the value in integerValue
Console.WriteLine("The integer value is " & integerValue)
```

Numeric Data Manip

Description of Code

Mathematical operations can be performed on numerical data, using either the variable holding the data.

There are a range of mathematical operations you can perform; the most comm

Symbol	Function	Example	Expla
+	Addition	X = 3 + 4	X would now store 7.
-	Subtraction	X = 5 – 2	X would now store 3.
*	Multiplication	X = 2 * 3	X would now store 6.
/	Division	X = 6 / 3	X would now store 2.
^	Exponential	X = 2 ^ 3	X would now store 2 to the p
MOD	Modulus	X = 10 MOD 5 X = 10 MOD 4	This keeps only the remainde 10/5 = 2, remainder 0. There 10/4 = 2.5. 4 * 2 = 8, so there Therefore 10 MOD 4 would re
\	Division	X = 5 \ 3	X stores the integer part of th 5/3 = 1.66 so 5\3 would retu

Code in

1. Stores 10 and 20 in two variables, adds them together and outputs the re

```
'store the integer 10 in the variable num1 and integer 20 i
Dim num1 As Integer = 10
Dim num2 As Integer = 20
Dim total As Integer

'add the values in num1 and num2, store the result in the v
total = num1 + num2

'output the calculation and result (total)
Console.WriteLine(num1 & " + " & num2 & " = " & total)
```

2. Stores 10 and 20 in variables, subtracts the value in num2 from the value

```
'store the integer 10 in the variable num1 and integer 20 i
Dim num1 As Integer = 10
Dim num2 As Integer = 20
Dim total As Integer

'subtract the value in num2 from num1, store the result in
total = num1 - num2

'output the calculation and result (total)
Console.WriteLine(num1 & " - " & num2 & " = " & total)
```

3. Stores 10 and 20 in variables, multiplies the values together and outputs t

```
'store the integer 10 in the variable num1 and integer 20 i
Dim num1 As Integer = 10
Dim num2 As Integer = 20
Dim total As Integer

'multiply the values in num1 and num2, store the result in
total = num1 * num2

'output the calculation and result (total)
Console.WriteLine(num1 & " * " & num2 & " = " & total)
```

4. Stores 1[illegible] 2[illegible] variables, divides the 10 by 20 and outputs the result

```
'store the integer 10 in the variable num1 and integer 20 i
Dim num1 As Integer = 10
Dim num2 As Integer = 20
Dim total As Single

'divide the value in num1 by num2, store the result in the
total = num1 / num2

'output the calculation and result (total)
Console.WriteLine(num1 & " / " & num2 & " = " & total)
```

5. Stores 10 and 3 in variables, calculates 10^3 and outputs the result.

```
'store the integer 10 in the variable num1 and integer 3 in
Dim num1 As Integer = 10
Dim num2 As Integer = 3
Dim total As Single

'calculate the value in num1 to the power of num2, store th
total = num1 ^ num2

'output the calculation and result (total)
Console.WriteLine(num1 & " ^ " & num2 & " = " & total)
```

6. Stores 10 and 3 in variables, calculates the modulus division of num1 MOD

```
'store the integer 10 in the variable num1 and integer 3 in
Dim num1 As Integer = 10
Dim num2 As Integer = 3
Dim total As Single

'calculate the modulus division of num1 MOD num2, store the
total = num1 Mod num2

'output the calculation and result (total)
Console.WriteLine(num1 & " Mod " & num2 & " = " & total)
```

7. Stores 10 and 3 in variables, calculates the integer division of num1 DIV n

```
'store the integer 10 in the variable num1 and integer 3 in
Dim num1 As Integer = 10
Dim num2 As Integer = 3
Dim total As Single

'calculate the integer division of num1 DIV num2, store the
total = num1 \ num2

'output the calculation and result (total)
Console.WriteLine(num1 & " \ " & num2 & " = " & total)
```

Selection: I

Description of Code

Selection statements let you run code depending on conditions. The code wil
condition is true, but will not be run if it is false. There are three levels of IF s
IF, IF ELSE and IF ELSEIF.

IF

```
If <condition> T
    <co      o  if condition is true>
End If
```

If the condition is true, then the code within the IF statement will run. If the c
statement is skipped and the program continues below the IF statement.

Code in Context

1. The program outputs *"The number is 10"* if the value in **theNumber** is equa

```
Dim theNumber As Integer = 10 'store the value 10 in the va
If theNumber = 10 Then 'if the value in the variable theNum
    MsgBox("The number is 10") 'output this message
End If
```

2. The program outputs *"Correct"* if th in **username** is equal to *"Bob12*

```
Dim username As St   = "Bob123" 'store "Bob123" in the va
If username   "  23" Then 'if the value in username is eq
    M     ("Correct") 'output this message
End If
```

3. The program adds 10 to **num1** if the value in **num1** is less than 10.

```
Dim num1 As Integer = 2 'store the value 2 in the variable
If num1 < 10 Then 'if the value in num1 is less than 10
      num1 = num1 + 10 'add 10 to the value in num1
End If
```

4. The program subtracts 10 from the value in **num1** if the value in **num1** is

```
Dim num1 As Integer = 2 'store the value 2 in the variable
If num1 > 10 Then 'if the value in num1 is greater than 10
      num1 = num1 - 10 'subtract 10 fro  e value in num1
End If
```

5. The program asks the us en a number; if this number is equal to 1

```
Dim us    mb    Integer
Dim nu    Integer = 10 'store the value 10 in the variable

'take as input a number, store it in the variable userNumbe
userNumber = InputBox("Enter a number")
If userNumber = num Then 'if the value in userNumber is equ
     MsgBox("Correct") 'output this message
End If
```

Selection: IF E

Description of Code

Selection statements let you run code depending on conditions. The code wil
condition is true, but will not be run if it is false.

There are three levels of IF statement: IF, IF ELSE and IF ELSEIF.

IF ELSE

```
If <condition> then
    <code to run if condition is true>
Else
    <code to run if condition is false>
End If
```

If the condition is true, then the code within the IF statement will run. If the c
code in the ELSE statement will run.

Code in Context

1. The program outputs "The number is 10" if the value in **theNumber** is equa
outputs "The number is not 10".

```
Dim theNumber As Integer = 10 'store the value 10 in the va
If theNumber = 10 Then 'if the value in theNumber is equal
     MsgBox("The number is 10") 'output this message
Else 'if the value in theNumber is not equal to 10
     MsgBox("The number is not 10") 'output this message
End If
```

2. The program outputs "Correct" if the value in **username** is equal to "Bob12
"That is incorrect".

```
Dim username As String = "Bob123" 'store the value Bob123 i
If username = "Bob123" Then 'if the value in username is eq
     MsgBox("Correct") 'output this message
Else 'if the value in username is not equal to Bob123
     MsgBox("That is incorrect") 'output this message
End If
```

3. The program adds 10 to the value in **num1** if the value in **num1** is less tha
from the value in **num1**.

```
Dim num1 As Integer = 2 'store the value 2 in the variable
If num1 < 10 Then 'if the value in num1 is less than 10
     num1 = num1 + 10 'add 10 to the value in num1, store t
Else 'if the value in num1 is not less than 10
     num1 = num1 - 10 'subtract 10 from the value in num1,
End If
msgbox(num1) 'output the value in num1
```

4. The program subtracts 10 from the value in **num1** if the value in **num1** is g
If not, it outputs "Too small".

```
Dim num1 As Integer = 2 'store the value 2 in the variable
If num1 > 10 Then 'if the value in the variable num1 is gre
     num1 = num1 - 10 'subtract 10 from t  alue in num1,
Else 'if the value in num1 is not g ea  chan 10
     MsgBox("Too small") 'ou   his message
End If
```

5. The pro sks the user to input a number. If that value is equal to the
it outputs "Correct". If not, it outputs "Incorrect".

```
Dim userNumber As Integer
Dim num As Integer = 10 'store the value 10 in the variable

'ask the user to enter a number, store it in the variable u
userNumber = InputBox("Enter a number")

'if the value in the variable userNumber is equal to the va
If userNumber = num Then
     MsgBox("Correct") 'output this message
Else 'if they are not equal
     MsgBox("Incorrect") 'output this message
End If
```

Selection: IF E

Description of Code

Selection statements let you run code depending on conditions. The code wil
condition is true, but will not be run if it is false.

There are three levels of IF statement: IF, ELSE a L F.

IF ELSEIF

```
If <condition> Then
    <code to run if condition is true>
ElseIf <condition> Then
    <code to run if this condition is true>
End If
```

If the condition is true, then the code within the If statement will run. If the c
condition will be checked; if this is true, the second set of code will run.

Any number of ELSEIFs can be added; for example:

```
If <condition> Then
    <code to run if condition is true>
ElseIf <condition> Then
    <code to run if this condition is true>
ElseIf <condition> Then
    <code to run if this condition is true>
ElseIf <condition> Then
    <code to run if this condition is true>
End If
```

In this example, if the first condition is false it will check the second, if this is f
etc. If one of the conditions is true, then the code within the condition will ru
will be checked.

This can also be combined with an ELSE. For example:

```
If <condition> Then
    <code to run if condition is true>
ElseIf <condition> Then
    <code to run if this condition is true>
ElseIf <condition> Then
    <code to run if this condition is true>
ElseIf <condition> Then
    <code to run if this condition is true>
Else
    <code to run if none of the conditions are
End If
```

There can only be one ELSE statement, which is last in the list; this will only r
conditions is true.

Code in Context

1. The program compares the value in **guess** to the value in **theNumber**. If t
“Correct”. If they are not equal, but **guess** is less than **theNumber**, it outp
equal, and **guess** is not less than **theNumber**, it outputs “Too large”.

```
Dim guess As Integer
Dim theNumber As Integer = 10 'the value 10 is stored in the v
'ask the user to input a number, stor d it in the variable gu
guess = InputBox("Guess the numb )
If guess = theNumber Then 'if the value in guess is equal to t
    MsgBox("Correct") 'output this message
'if not and the value in guess is less than the value in t
ElseIf  < theNumber Then
    Msg  "Too Small") 'output this message
Else 'if neither condition is true
    MsgBox("Too large") 'output this message
End If
```

2. The program asks the user to input a score. If the value in **score** is greate
“Brilliant, well done”. If not, it checks whether the value is greater than o
program outputs “Fab, you did really well”. The program continues check
If the value does not meet any of the criteria, it outputs “Oh dear, some e

```
Dim score As Integer
'ask the user to input a score, store it in the variable score
score = InputBox("Enter your score")
If score >= 90 Then 'if the value in score is greater than or
     MsgBox("Brilliant, well done") 'out  h s message
ElseIf score >= 80 Then 'if not, and if re is greater than
     MsgBox("Fab, you did rea  ll ) 'output this message
ElseIf score >= 70 Then  and if score is greater than
     MsgBox("That  y good!") 'output this message
ElseIf  e  hen 'if not, and if score is greater than
     M  "t bad, but I think you can do better") 'output
ElseIf  >= 50 Then 'if not, and if score is greater than
     MsgBox("You got at least half marks; you can improve on t
ElseIf score >= 40 Then 'if not, and if score is greater than
     MsgBox("Not quite half marks; need to try harder") 'outpu
Else 'if none of the conditions is true
     MsgBox("Oh dear, some extra work needed here") 'output th
End If
```

3. The program asks the user to input a subject. If the value in **subject** is eq
it outputs “Good choice”. If not, it compares it to “Maths”, “French” and “P
any, it outputs “Is that even a subject?”.

```
Dim subject As String
'ask the user to enter a subject and st  in the variable
subject = InputBox("Enter your fav  t  bject")
If subject = "Computer Scie  " Then 'if the value in subject
    MsgBox("Good choi  ") 'output this message
ElseIf subject  "M  " Then 'if not, and subject is equal to
    Ms  "W  100/2*6-2*2.698?") 'output this message
ElseIf  ct = "French" Then 'if not, and subject is equal t
    Msg  "Très bien") 'output this message
ElseIf subject = "Physics" Then 'if not, and subject is equal
    MsgBox("Why does Earth go around the Sun?") 'output this m
Else 'if none of the conditions is true
    MsgBox("Is that even a subject?") 'output this message
End If
```

Code in Context

4. The programs tells the user to enter a number. If the number is less than
   If not, but it is less than 25, it adds 5 to it. If it is not less than 25, it subt
   The program then outputs the value in **numEntered**.

```
Dim numEntered As Integer

'ask the user to input a number, st re it in the variable n
numEntered = InputBox("Enter  be  )
If numEntered < 10 Ther  t  value in numEntered is less

    'add 10 t  value in numEntered, store it in the var
    nu rt  numEntered + 10
ElseIf ntered < 25 Then 'if not, and numEntered is less
    numEntered = numEntered + 5 'add 5 to numEntered, store
Else 'if none of the conditions is true

    'subtract 2 from numEntered, store result in numEntered
    numEntered = numEntered - 2
End If
MsgBox(numEntered) 'output the value in numEntered
```

Selection: Select

Description of Code

Selection statements let you run code depending on conditions. In a Select Ca
uses one variable and can compare it to a number of optio . If the condition
will run. If none of the conditions is true, an Else (if) can be run.

The code:

```
Select Case(<variable>)
Case <value>
     <code to run if condition is true>
Case <value>
     <code to run if condition is true>
Case Else
     <code to run if none of the conditions are
End Select
```

You can have any number of Cases in a Select Case statement, but only one Ca

You can have multiple conditions under one Case condition. These are treated
have a comma between each value (see example 2). In the following example
value2, then the code will run:

```
Select Case(<variable>)
Case <value1>,
     <co    o    if condition is true>
End Sele
```

Ranges

You can use a Select Case to check ranges of values:

```
Select Case(<variable>)
Case Is <condition>
      <code to run if condition is true>
End Select
```

Code in Context

1. The program asks the user to enter a number and stores this in the variable n
value in the variable **number** is 0; if it is, it outputs "Zero". If it is not 0, it che
outputs "One". If it is not 0 or 1, if checks whether it is 2; if it is, it outputs "T
whether it is 3; if it is, it outputs "Three". If it is not 0, 1, 2 or 3, it outputs "No

```
Dim number As String
Console.WriteLine("Enter a number") 'ask the user to enter a
number = Console.ReadLine() 'store the number entered in the

Select Case number 'compare the value in number
    Case 0 'if number is 0
        Console.WriteLine("Zero") 'output this message
    Case 1 'if number is 1
        Console.WriteLine("One") 'output this message
    Case 2 'if number is 2
        Console.WriteLine("Two") 'output this message
    Case 3 'if number is 3
        Console.WriteLine("Three") 'output this message
    Case Else 'if none of the case comparisons is met
        Console.WriteLine("Not zero, one, two or three") 'o
End Select
```

2. The user is asked to enter an animal; this is stored in the variable **animal**.
compared to "Bird"; if it is equal, the program outputs "This animal has 2 l
whether it is "Horse", "Dog", "Cat" or "Rabbit"; if it is any of these, it output
is not, it checks whether it is "Snake" or "Worm" and if so, it outputs "This an
checks whether it is "Centipede"; if it is it outputs "This animal has lots o
options, then it outputs "Sorry I don't recognise that animal".

```
Dim animal As String

'ask the user to input an animal, store it in the variable a
animal = InputBox("Enter an animal")

Select Case animal 'compare the value in animal
    Case "Bird" 'if animal is equal to Bird
       MsgBox("This animal has 2 legs") 'output this messag

    'if animal is equal to Horse, or Dog, or Cat, or Rabbit
    Case "Horse", "Dog", "Cat", "Rabbit"
        MsgBox("This animal has 4 legs") 'output this messa
    Case "Snake", "Worm" 'if animal is equal to Snake or Wo
        MsgBox("This animal has 0 legs") 'output this messa
    Case "Centipede" 'if animal is equal to Centipede
        MsgBox("This animal has lots of legs") 'output thi
    Case Else 'if none of the case comparisons is met
        MsgBox("Sorry I don't recognise that animal") 'outp
End Select
```

3. The program asks the user to enter a number and stores it in the variable n
10, the program outputs "It's less than 10". If not, it checks whether it is e
10". If not, it checks whether it is greater than 10; if it is, it outputs "It's m

```
Dim number As Integer

'ask the user to input a number, store it in the variable n
number = InputBox("Enter a n...")

Select Case numb... 'compare using the value in number
    Cas... 'if the value in number is less than 10
        Box("It's less than 10") 'output this message
    Ca... 'if the value in number is equal to 10
        MsgBox("It's 10") 'output this message
    Case Is > 10 'if the value in number is greater than 10
        MsgBox("It's more than 10") 'output this message
End Select
```

4. The program asks the user to enter their age and stores it in the variable a
is less than or equal to 12; if it is, it outputs "You are a child". If not, it che
equal to 19; if it is, it outputs "You are a teenager". If not, it checks wheth
if it is, it outputs "You are in your 20s". If not, it checks whether it is less t
outputs "You are in your 30s". If none of these cases is true, then it output

```
Dim age As Integer
Console.WriteLine("Enter your age") 'output this message
age = Console.ReadLine() 'store the value the user enters i

Select Case age 'compare using the value in age
    Case <= 12 'if age is less than or equal to 12
        ...WriteLine("You are a child") 'output this m
    Ca... 19 'if not, and age is less than or equal to 19
        ...sole.WriteLine("You are a teenager") 'output thi
    Case <= 29 'if not, and age is less than or equal to 29
        Console.WriteLine("You are in your 20s") 'output th
    Case <= 30 'if not, and age is less than or equal to 30
        Console.WriteLine("You are in your 30s") 'output th
    Case Else 'if none of the conditions is true
        Console.WriteLine("You're getting old") 'output thi
End Select
```

Operators: Relat

Description of Code

Relational operators are used in comparisons; for example, in selection (IF) an
They are used in expressions which return either true o e.

Operator	Description
<	**Less than** *Is the value to the left of the operator less than the value to the right*
>	**Greater than** *Is the value to the left of the operator bigger than the value to the right?*
<=	**Less than or equal to** *Is the value to the left of the operator less than, or equal to, the value to th*
>=	**Greater than or equal to** *Is the value to the left of the operator bigger than, or equal to, the value to*
=	**Equal to** *Is the value to the left of the operator equal to the value to the right?*
<>	**Not equal to** *Is the value to the left of the operator not equal t value to the right?*

Code in Context

1. The pro sks the user to enter two numbers; it then outputs the larg

```
Dim num1, num2 As Integer
Console.WriteLine("Enter a number") 'ask the user to enter
num1 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a second number") 'ask the user to
num2 = Console.ReadLine 'read the value and store it in the

If num1 < num2 Then 'if the value in num1 is less than the
    Console.WriteLine(num2) 'output the value in num2
Else 'otherwise
     Console.WriteLine(num1) 'output the value in num1
End If
```

2. The program asks the user to enter two number , he outputs the sma

```
Dim num1, num2 As Integer
Console.WriteLine("Enter a number") 'ask the user to enter
num1 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a second number") 'ask the user to
num2 = Console.ReadLine 'read the value and store it in the

If num1 > num2 Then 'if the value in num1 is greater than t
    Console.WriteLine(num2) 'output the value in num2
Else 'otherwise
    Console.WriteLine(num1) 'output the value in num1
End If
```

3. The program asks the user to enter two numbers. It outputs the larger of t
the same, it outputs "Same".

```
Dim num1, num2 As Integer
Console.WriteLine("Enter a number") 'ask the user to enter
num1 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a second number") 'ask the user to
num2 = Console.ReadLine 'read the value and store it in the

If num1 > num2 Then 'if the value in num1 is greater than t
    Console.WriteLine(num1) 'output the value in num1
ElseIf num1 < num2 Then 'if the value in num1 is less than
    Console.WriteLine(num2) 'output the value in num2
Else 'if neither if condition is true
    Console.WriteLine("Same") 'output "Same"
End If
```

4. The program asks the user to enter two numbers. It outputs "Same" if they

```
Dim num1, num2 As Integer
Console.WriteLine("Enter a number") 'ask the user to enter
num1 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a second number") 'ask the user to
num2 = Console.ReadLine 'read the value and store it in the

If num1 = num2 Then 'if the value in num1 is equal to the v
    Console.WriteLine("Same") 'output "Same"
Else 'otherwise
    Console.WriteLine("Different") 'output "Different"
End If
```

5. The program asks the user to enter two numbers. It outputs "Different" if
or "Same" if they are equal.

```
Dim num1, num2 As Integer
Console.WriteLine("Enter a number") 'ask the user to enter
num1 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a second number") 'ask the user to
num2 = Console.ReadLine 'read the value and store it in the

If num1 <> num2 Then 'if the value in num1 is equal to the
    Console.WriteLine("Different") 'output "Different"
Else 'otherwise
    Console.WriteLine("Same") 'output "Same"
End If
```

Operators: Bool

Description of Code

Boolean operators are used in comparisons; for example i[n] selection (IF) and i[n]

They are used in expressions which return either tr[ue] or false. AND and OR ta[ke]
and determine whether the result is t[rue] o[r] false.

Operator	Description	
And	**Logical AND** All conditions must be true for the outcome to be true.	2 < 2 < 10
Or	**Logical OR** At least one condition must be true for the outcome to be true.	2 < 10 < 2 < 10
Not()	**Logical NOT** If the statement in the brackets is true, return false. If the statement in the brackets is false, return true.	No No

Code in Context

1. The program asks the user to enter [fo]ur numbers. If the 1st number is less
is less than the 4th, it adds [it] to get[her] the 1st and 3rd values. If not, it adds t[he]

```
Dim num[1], num2, num3, num4, total As Integer
Console.WriteLine("Enter a number") 'ask the user to enter a
num1 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a second number") 'ask the user to
num2 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a third number") 'ask the user to e
num3 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a fourth number") 'ask the user to
num4 = Console.ReadLine 'read the value and store it in the

'if the value of num1 is less than num2, AND the value of n
If num1 < num2 And num3 < num4 Then
    total = num1 + num3 'the value in total becomes the val
Else   'if not
    total = num2 + num4 'the value in total becomes the val
End If
Console.WriteLine(total) 'output the value in the variable
```

2. The program asks the user to enter four numbers. If the 1st number is less number is less than the 4th, it adds together the 1st and 3rd values. If not,

```
Dim num1, num2, num3, num4, total As Integer
Console.WriteLine("Enter a number") 'ask the user to enter
num1 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a second number") 'ask the user to
num2 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a third number") 'ask the user to
num3 = Console.ReadLine 'read the value and store it in the
Console.WriteLine("Enter a fourth number") 'ask the user to
num4 = Console.ReadLine 'read the value and store it in the

'if the value in num1 is less than num2, OR the value in nu
If num1 < num2 Or num3 < num4 Then
    total = num1 + num3 'the value in total becomes the val
Else   'if not
    total = num2 + num4 'the value in total becomes the val
End If
Console.WriteLine(total) 'output the value in the variable
```

3. The program asks the user to enter two numbers. If both numbers are gre outputs "You passed both". If not, but one of the numbers is greater than "You passed one". If neither is greater than or equal to 60, it outputs "You

```
Dim mark1, mark2 As Integer
Console.WriteLine("Enter the first mark") 'ask the user to
mark1 = Console.ReadLine 'read the input and store it in th
Console.WriteLine("Enter the second mark") 'ask the user to
mark2 = Console.ReadLine() 'read the input and store it in

'if the value in mark1 is greater than or equal to 60 and t
greater than or equal to 60
If mark1 >= 60 And mark2 >= 60 Then
    Console.WriteLine("You passed both") 'output the messag

'if the value in mark1 is greater than or equal to 60, or t
greater than or equal to 60
ElseIf mark1 >= 60 Or mark2 >= 60 Then
    Console.WriteLine("You passed one") 'output the message
Else 'if neither condition is true
    Console.WriteLine("You didn't pass either") 'output the
End If
```

4. The program asks the user to input a subject. If the value is not "Comput How can that be?".

```
Dim subject As String
Console.WriteLine("What is your favourite subject?") 'ask u
subject = Console.ReadLine 'store the input in the variable

'if the value in the variable subject is not equal to "Comp
If Not (subject = "Computer Science") Then
    Console.WriteLine("What!! How can that be?") 'output th
End If
```

5. The programs asks the user to input a number and stores it in **num1**. It lo
is not equal to 10. Within each iteration, it adds 1 to the value in **num1**.

```
Dim num1 As Integer

Console.WriteLine("Enter a number") 'a
num1 = Console.ReadLine 'read th
While (Not (num1 = 1
    num1 = num1
End Wh
```

6. The program asks the user to enter two numbers. It loops until either of t
not true. It counts the number of times it loops, and adds 1 to the value i
outputs the number of times it runs.

```
Dim num1, num2, count As Integer
Console.WriteLine("Enter the first number") 'ask the user t
num1 = Console.ReadLine 'read the input and store it in the
Console.WriteLine("Enter the second number") 'ask the user
num2 = Console.ReadLine 'read the input and store it in the

count = 0 'set the value in count to 0

'loop while both the value in num1 is less than 10 and the
than the value in num2 are false
While (Not (num1 > 10 And num2 > nu
    num1 = num1 + 1 'add 1 t
    count = count + 1
End While
Conso
```

String Manipulation

Description of Code

We can find out the length of a string, either in a variable or within quotes ("")

The code:

<string>.length

returns the number of characters within the string as an integer.

Code in Context

1. Store "Hello World" in a variable and output the number of characters in t

```
Dim newString As String
newString = "Hello World" 'store "Hello World" in the varia
Console.WriteLine(newString.Length) 'output the length of t
```

2. Count the number of characters in "This is a sentence" and output the res

```
Dim words As String
Dim wordsLength As Integer
words = "This is a sentence" 'store "This is a sentence" in

'store the number of characters from the variable words in
wordsLength = words.Length

'output the text plus the value in the variable wordsLength
Console.Write "The sentence has " & wordsLength & " cha
```

3. Ask the user to input a colour, then output the numbers from 0 to the num entered.

```
Dim colour As String
Console.WriteLine("Enter your favourite colour")'ask the us
colour = Console.ReadLine 'store the input in the variable

'loop from 0 to the number of characters in the variable co
For count = 0 To Len(colour) - 1
    Console.WriteLine(count) 'output the value in count
Next count
```

4. Ask the user to input a four-letter word. If it's not four letters, tell them; ot

```
Dim userInput As String
Console.WriteLine("Enter a four letter word")

'ask the user to input a 4-letter word, save it in the vari
userInput = Console.ReadLine
If userInput.Length <> 4 Then 'if the length of the word en
    Console.WriteLine("Sorry that's not a 4-letter word.")
Else     'otherwise (the length is 4)
    Console.writeline("That was a good word") 'output the m
End If
```

String Manipulation: S

Description of Code

You can extract specific characters from within a string: either a number of ch
left, a number of characters from the right or at a specific point in the string.

Microsoft.VisualBasic.Left(<stri g , umber of lett
returns the number of letters requested e left of the string (starting at

Microsoft.VisualP ght(<string>,<number of let
returns the r r requested from the right of the string (starting a

Microsof ualBasic.Mid(<string>,<starting letter
returns the number of letters requested from the left of the string (starting at

Code in Context

1. Store "Hello World" in a variable, then extract and output "Hello" from it.

```
Dim newWords As String
newWords = "Hello World" 'store "Hello World" in the variable
'output the first 5 characters in the variable newWords
Console.WriteLine(Microsoft.VisualBasic.Left(newWords, 5))
```

2. Output the first 10 characters that the user inputs.

```
Dim theInput, extract As String
Console.WriteLine("Type a messag e user to input a m
theInput = Console.ReadLin e the input in the variable
'output the last 10 c rs in the message, store them in t
extract = Mic /isualBasic.Right(theInput, 10)
Consol e( The last 10 characters are " & extract) 'outp
```

3. Output the first half of the characters the user inputs.

```
Dim theInput, extract As String
Dim inputLength As Integer
Console.WriteLine("Type a message") 'ask the user to input a m
theInput = Console.ReadLine 'store the input in the variable t
'count the number of characters in theInput, divide it by 2, s
inputLength = Int(theInput.Length / 2)
'count the number of characters in theInput and divide it by 2
'nearest integer, then store it in inputLength
extract = Microsoft.VisualBasic.Left(theInput, inputLength)
'output the text and the value in extract
Console.WriteLine("The first half of the m e is " & extract
```

4. Output each character from a stri haracter at a time.

```
Dim theInput As Str
Dim input ngt nteger
Consol te ine("Type a message") 'ask the user to input a m
theInp onsole.ReadLine 'store the input in the variable t
inputLength = theInput.Length 'count number of characters in th
For x = 1 To inputLength 'loop from 0 to the number of charact
    'output the letter at position x in the string in theInput
    Console.WriteLine(Microsoft.VisualBasic.Mid(theInput, x, 1)
Next x
```

String Manipulatio

Description of Code

A string can be turned into lower case or into upper case.

The code:

UCase(<string>)

will turn each letter in th[illegible] upper case. Characters that are not letters

The code:

LCase(<string>)

will turn each letter in the string to lower case. Characters that are not letters

Code in Context

1. Output "Hello World" in all lower case, then all upper case.

```
Dim theText, upperCase, lowerCase As String

theText = "Hello World" 'store "Hello Wor[illegible] in the variabl

'convert the contents of theText to [illegible]case, store in uppe
upperCase = UCase(theTex[illegible]

'convert the co[illegible] theText to lower case, store in lowe
lowerCa[illegible] = [illegible]Case(theText)

'outpu[illegible] contents of lowerCase, a space, and then the co
Console.WriteLine(lowerCase & " " & upperCase)
```

2. Convert the first half of a string to lower case and the second half to uppe

```
Dim inputText, firstHalf, secondHalf As String
Dim lengthText, midText As Integer
Console.WriteLine("Enter a message") 'ask the user to input
inputText = Console.ReadLine 'store it in inputText

'count number of characters in inputText, store in lengthTex
lengthText = inputText.Length

'halve number of letters, round to inte[illegible] store in midTex
midText = Int(lengthText / 2)

'convert the first half [illegible]Text to lower case, store i
firstHalf = LCase(Mi[illegible]sof[illegible].VisualBasic.Left(inputText, mid

'conver[illegible]e [illegible]d half of inputText to upper case, store
second[illegible] = UCase(Microsoft.VisualBasic.Right(inputText, le

'output the values in firstHalf and secondHalf
Console.WriteLine(firstHalf & secondHalf)
```

String Manipulat
Concatenatio

Description of Code

Concatenation means joining two strings together to become one string.

The code:

```
<strin  tring>
```

joins the two strings together to form one string.

Code in Context

1. Join "Hello" and "World" with a space to become "Hello World".

```
Dim first, second, message As String
first = "Hello" 'store Hello in first
second = "World" 'store World in second

'store the contents of first, a space, and then the content
message = first & " " & second
Console.WriteLine(message) 'output the contents of message
```

2. Ask the user to enter their first name and surname; then output "Hello" fo
then surname.

```
Dim fi me, surname As String
Console.WriteLine("Enter your firstname") 'ask the user to
firstname = Console.ReadLine 'store the input in firstname
Console.WriteLine("Enter your surname") 'ask the user to en
surname = Console.ReadLine 'store the input in surname

'output Hello, followed by the contents of firstname, a spa
'contents of surname
Console.WriteLine("Hello " & firstname & "  " & surname)
```

3. Ask the user to input a colour and an animal. Concatenate the colour and
output it in a sentence.

```
Dim colour, animal, final As String
Console.WriteLine("Enter your fa  it  colour") 'ask the u
colour = Console.ReadLine() s  the input colour
Console.WriteLine("E  r y  favourite animal") 'ask the u
animal = Consol  ne() 'store the input animal

'stor  contents of colour, a space, and then the conten
'varia  final
final = colour & " " & animal

'output the text with the contents of final
Console.WriteLine("A " + final + " is an interesting animal
```

String Manipulat
Type Check

Description of Code

You can find out whether a strin[illegible]ific data type; for example, whether whether it is all lower ca[illegible]ether it is all upper case.

The code:

isnumeric(<string>)

returns true if it is numeric, and false if it is not.

Code in Context

1. Ask the user to input characters. If they are all numbers, output "It's all n "It's not all numbers".

```
Dim textInput As String
Console.WriteLine("Enter characters")

'ask the user to input characters, store in textInput
textInput = Console.ReadLine
If IsNumeric(textInput) Then 'if all the characters
    Console.Writ[illegible]s all numbers") 'output the messa
Else [illegible]
    C[illegible].WriteLine("It's not all numbers") 'output the m
End If
```

String Manipulatio

Description of Code

A character can be turned into its ASCII code, and an ASCII code can be turned

The code:

ASC(<single charact

returns the ASCII numb[illegible] character.

The code:

CHR(<ASCII number>)

returns the character of the ASCII number.

Code in Context

1. Output the ASCII value of "?".

```
Dim letter As Char
letter = "?" 'store ? in letter
Console.WriteLine(Asc(letter)) 'output the ASCII value of t
```

2. Output the ASCII value of a character the user inputs.

```
Dim letter As Char
Console.WriteLine("Enter a character") 'ask the user to ent
letter = Console.ReadLine() 'store the input in letter
Console.WriteLine(Asc(letter)) 'output the ASCII value of t
```

3. Output the character for the number the user inputs.

```
Dim numInput As Integer
Console.WriteLine("Enter a number") 'ask the user to enter
numInput = Console.ReadLine() 'store the input in numInput
Console.WriteLine(Chr(numInput)) 'output the character for
```

4. Ask the user to input a sentence. Output the ASCII value of each characte

```
Dim letterInput As String
Dim letterNum As Integer
Console.WriteLine("Enter a sentence") 'ask the user to ente
letterInput = Console.ReadLine 'read the input and store it
For x = 1 To Len(letterInput) 'loop from 1 to the length of
    'turn character number x into ASCII, store in letterNum
    letterNum = Asc(Microsoft.VisualBasic.Mid(letterInput, x
    Console.WriteLine(letterNum) 'output the value in lette
Next x
```

Iteration: FO

Description of Code

A FOR loop is a count-controlled loop; you need to know how many times it will r

```
For <variable> = <start value> t  d value>
    <statements to repeat>
Next <variable>
```

The variable act s a it is initialised at the start value, it then runs the sta command in th counter by 1. The program then moves back to the FOR a counter to the and end values. If it is still within these bounds, it runs the sta 1 before going back to the start. This keeps on repeating until the value in the cou

For example:

```
For counter = 0 to 3
    Console.writeline(counter)
Next counter
```

- This code will start by initialising counter to 0. It runs the code inside the counter, 0). Next counter means counter is now 1.
- It goes back to the For statement and checks whether counter is between code inside the loop (outputs 1). Next counter increases counter to 2.
- It goes back to the For statement and checks wh ounter is between code inside the loop (outputs 2). Next cou ter eases counter to 3.
- It goes back to the For statem nd hecks whether counter is between code inside the loop (ts 3). Next counter increases counter to 4.
- It goes b k t statement and checks whether counter is between skips de inside the loop, skips the Next counter line and continues w

Adjusting the increment

The NEXT statement does not have to increase by 1 each time; you can set the value i

```
For <variable> = <start value> to <end value> Step
    <statements to repeat>
Next <variable>
```

Going down. The following code would start counter at 3, and then decrease the valu

```
For counter = 3 to 0 Step -1
    Console.writeline(counter)
Next counter
```

Decimal increments

The following co wo counter at 2, and then increase the value in counter by

```
For coun = 2 to 3 Step 0.1
    Console.writeline(counter)
Next counter
```

Code in Context

1. Output the numbers 0, 1, 2, 3.

```
For counter = 0 To 3 'set counter to start at 0, loop until i
    Console.WriteLine(counter) 'output the value in counter
Next counter 'increase the value in counter by 1
```

2. Display the times table (up to 12 times the number) for a number the use

```
Dim limit As Integer
Dim result As Integer

'ask the user to enter a number
Console.WriteLine("Enter the number of the times table you
limit = Console.ReadLine 'read the input and store in the v
For counter = 0 To 12 'set counter to start at 0, loop until i
    result = counter * limit 'calculate counter multiplied
    Console.WriteLine(counter & " * " & limit & " = " & res
Next counter 'increase the value in counter by 1
```

3. Ask the user to input a number; output that many "*"s on the same line.

```
Dim userInput As Integer
Dim message As String

'ask the user to enter a number
Console.WriteLine("Enter the number of *s you want displaye
userInput = Console.ReadLine 'read the input and store in t

'set x to start at 0, loop until it is outside the range 0
For x = 1 To userInput
    message = message & "*"
Next x 'increase the value in x by 1
Console.WriteLine(message)
```

4. Output a countdown from 10 to 1, then display "Blast Off!".

```
For count = 10 To 1 Step -1 'loop from 10 to 1, setting the
     Console.WriteLine(count) 'output the value in count
Next count 'decrease count by 1
Console.WriteLine("Blast Off!!") 'display the message
```

5. Output the numbers 1 to 2, increasing by 0.1 each time.

```
For count = 1 To 2 Step 0.1 'loop from 1 to 2 increasing by
    Console.WriteLine(count) 'output the value in count
Next count 'increase count by 0.1
```

6. Output alternate numbers from 0 to the number the user inputs.

```
Dim stopValue As Integer
Console.WriteLine("Enter the value to stop at") 'ask the use
stopValue = Console.ReadLine 'read the input and store in s
For count = 0 To stopValue Step 2 'loop from 0 to stopValue,
    Console.WriteLine(count) 'output the value in count
Next count 'increase count by 2
```

Iteration: Whi

Description of Code

A WHILE loop is a condition-controlled loop; it is usually used when you do no
times the loop will run, although it can also be used as a count-controlled loo

It loops (and continues looping) while a conditio is True. When the condition
False, it stops looping.

```
While <condition>
    <statements to repeat>
End While <variable>
```

Code in Context

1. Output the numbers from 0 to 10.

```
Dim counter As Integer
counter = 0 'store 0 in the variable counter
While counter <= 10 'loop while counter is less than or equ
    Console.WriteLine(counter) 'output the value in the var
    counter = counter + 1 'add 1 to the valu in counter
End While
```

2. Loop asking for input while the user enters "Y".

```
Dim in  l  string = "Y" 'store "Y" in inputValue

'loop  value in inputValue as an upper case equals "Y"
While UCase(inputValue) = "Y"
    Console.WriteLine("Enter Y to continue or N to stop") '
    inputValue = Console.ReadLine() 'read the user input, s
End While
Console.WriteLine("OK, we've stopped") 'output the message
```

3. Generate and output random numbers between 0 and 100 until a number

```
'call Random class in preparation for generating random num
Dim randomclass As New Random()
Dim randomNumber As Integer = 0 'store 0 in randomNumber

While randomNumber <= 50 'while the va  in randomNumber is

    'generate a random  tween 0 and 100, store in r
    randomNumber =  omclass.Next(0, 101)
    Console.W  ne(randomNumber) 'output the value in ra
End Wh
```

4. Ask the user to input numbers until they do not want to continue. Count h
   Calculate the mean average of all the numbers entered. Output the large
   smallest number entered.

```
Dim userNum, counter, total As Integer
Dim userContinue As String = "Y" 'stor[illegible] in userContinue
Dim largest As Integer = 0 'store 0 in largest
Dim smallest As Integer = 9999 'store 9999 in smallest

'loop while the value in userContinue is one of the options
While userContinue = "Yes" Or userContinue = "Y" Or userCon
userContinue = "yes"
    Console.WriteLine("Enter the first number") 'output the
    userNum = Console.ReadLine 'read the user input, store
    total = total + userNum 'add userNum to the running tot
    counter = counter + 1 'add 1 to the value in counter

    'check if the input is the largest value so far.
    'if the value in userNum is larger than the value in la
    If userNum > largest Then
        largest = userNum 'store the value in userNum in la
    End If

    'check if the input is the smallest value so far
    'if the value in userNum is smaller than the value in sma
    If userNum < smallest Then
        smallest = userNum 'store the val[illegible]n userNum in s
    End If

    Console.WriteLine("Continue") 'output "Continue"?
    userContinue = Console.ReadLine 'read the user input an
End While

'output the count of numbers entered
Console.WriteLine("You entered " & counter & " numbers")
Console.WriteLine("The largest number is " & largest) 'outp
Console.WriteLine("The smallest number is " & smallest) 'ou

'calculate and output the average of all the numbers input
Console.WriteLine("The mean average of the numbers is " & (
```

Iteration: Repeat

Description of Code

A Repeat Until loop is a condition-controlled loop; it is usually used when you
many times the loop will run, although it can also be
a count-controll

It loops (and continues looping) until
tion is true. When the condition
it stops looping.

The loop can
he condition either first or last. Testing the condition last
loop will alwa
run once before the condition is tested.

Testing the condition first:

```
Do Until <condition>
    <statements to repeat>
Loop
```

Testing the condition last:

```
Do
    <statements to repeat>
Loop Until <condition>
```

Code in
ext

1. Output the numbers 0 to 10.

```
Dim counter As Integer = 0 'store 0 in counter
Do Until counter > 10 'loop until the value in counter is g
    Console.WriteLine(counter) 'output the value in counter
    counter = counter + 1 'add 1 to the value in counter
Loop
```

2. Output the numbers 0 to 10.

```
Dim counter As Integer = 0 'store 0 in
Do
    Console.WriteLine(
) output the value in counter
    counter = counte
1 add 1 to the value in counter
Loop Until co
> 10 'loop until the value in counter is
```

3. Ask the user if they want another slice of pizza until they say "No". Output they had.

```
Dim answer As String
Dim numSlices As Integer = 0 'store 0 in numSlices
Console.WriteLine("Here's a slice of pizza") 'output the me

Do
    numSlices = numSlices + 1 'add 1 to the value in numSli
    Console.WriteLine("Would you another slice of pizza?")
    answer = Console.ReadLine() 'read the user input and st

'loop until the upper-case value of the string entered equa
Loop Until (UCase(answer) = "NO" Or UCase(answer) = "N")

'output the number of slices of pizza the user chose
Console.WriteLine("You had " & numSlices & " slices of pizz
```

4. Ask the user to enter their age until it is a given value (in this case, greater than or equal to 130).

```
Dim age As Integer

Do
    Console.WriteLine("Enter your age in years") 'output th
    age = Console.ReadLine() 'read the user input and store

    'if the value in age is less than 0 or greater than 130
    If age < 0 Or age > 130 Then

        'output the message
        Console.WriteLine("That doesn't seem like a valid a
    End If

'loop and continue until the value in age is greater than o
'than or equal to 130
Loop Until age >= 0 And age <= 130
```

1D Arrays

Description of Code

An array is a structure that can store many pieces of data, unlike a variable, wh
one piece of data. An array has one identifier (name) and then a number of sp
'elements') where pieces of data can be put. An array can only store one type o
declared as a string, it can only store str

An array can be visualised as a table, for example:

Index	0	1	2	
Data	"Red"	"Blue"	"Orange"	

Arrays start counting at 0. This array has five elements: index 1 is Blue, index

Declaration

Like a variable, an array needs to be declared.

```
Dim <identifier>(<start number> to <end number>)
```

For example, to declare an array named colours with five elements that store

```
Dim colours(0 to 4) As String
```

Adding data

Data can be added into a specific array element:

```
<identifier>(<index>) = <data>
```

For example, to store "Blue" in elem colours:

```
colours(1) = "Bl
```

Accessing da

Data in a spec array element can be accessed:

```
<identifier>(<index>)
```

For example, to access the data in colours in index 1:

```
colours(1)
```

Code in Context

1. Store five colours in a list and output each element in the array.

```
Dim colours(0 To 4) As String 'declare a r ay with 5 elem

colours(0) = "Red" 'store  lement 0
colours(1) = "Blue" 'st ore lue in element 1
colours(2) = "O  store Orange in element 2
colour  = "Yellow" 'store Yellow in element 3
colour  = "Purple" 'store Purple in element 4

'output each element in the array
For x = 0 To 4 'loop from the first element (0) to the last
    Console.WriteLine(colours(x)) 'output the value in elem
Next x 'increase x by 1
```

2. Ask the user to enter five colours; store each one in the array, and then ou

```
Dim colours(0 To 4) As String 'declare an array with 5 elem

For x = 0 To 4 'loop from the first element (0) to the last
    Console.WriteLine("Enter a colour") 'output the message
    colours(x) = Console.ReadLine 'read the user input, sto
Next x

'output each element in the array
For x = 0 To 4 'loop from the first element (0) to the last
    Console.WriteLine(colours(x)) 'output the value in elem
Next x 'increase x by 1
```

3. Store the months of the year in an array. Ask the user to input a month n of that month.

```
Dim months(0 To 11) As String 'declare an array named month
Dim userInput As Integer

'store each month in its own array element
months(0) = "January"
months(1) = "February"
months(2) = "March"
months(3) = "April"
months(4) = "May"
months(5) = "June"
months(6) = "July"
months(7) = "August"
months(8) = "Septe
months(9) =
months(10) = "November"
months(11) = "December"

Console.WriteLine("Enter the month number") 'output the mes
userInput = Console.ReadLine - 1 'read the user input, subtr

Console.WriteLine(months(userInput)) 'output the data in mont
```

4. Take 10 numbers from the user and store in the array userNumbers. Outp element the user inputs.

```
'declare an array named userNumbers with 10 elements that a
Dim userNumbers(0 To 9) As Integer

'read a number into each array elem
For counter = 0 To 9 'loop from the first element in counte
    Console.WriteLine("Enter a number between 1 and 10") 'o

    'read the user input and store it in userNumbers as pos
    userNumbers(counter) = Console.ReadLine
Next counter 'increase counter by 1

Console.WriteLine("Which number do you want to check?") 'ou

'read the user input, subtract 1, access that element in us
Console.WriteLine(userNumbers(Console.ReadLine - 1))
```

5. Generate 100 random numbers between 1 and 1,000. Ask the user what n
Search the array for that number, and, if it finds it, output the array eleme

```
Dim randomclass As New Random() 'declare randomclass for ge

'declare an array named randomNumbers wit  0 elements tha
Dim randomNumbers(0 To 99) As Integ
Dim userInput As Integer

'generate 100 rand  mbers
For x = 0 To  p from array element 0 to element 99

    'g  e a random number between 1 and 1000, store in ra
    ra mNumbers(x) = randomclass.Next(1, 1001)
Next x 'increment x

Console.WriteLine("What number would you like to find?") 'o
userInput = Console.ReadLine 'read the user input and store

'check each element to find out if it's the same as userInp
For count = 0 To 99 'loop from array element 0 to element 9

    'if array at element count is equal to userInput
    If randomNumbers(count) = userInput Then

       'output element number where it was found
       Console.WriteLine("Found in position " & count) '
       count = 100 'break the loop by setting count to be o
    End If
Next count 'increment count
```

6. Generate 100 random between 1 and 100. Add together all the
the total

```
Dim ra lass As New Random() 'declare randomclass for ge

'declare an array named randomNumbers with 100 elements tha
Dim randomNumbers(0 To 99) As Integer
Dim total As Integer

'generate 100 random numbers
For x = 0 To 99 'loop from array element 0 to element 99

    'generate a random number between 1 and 1000, store in ra
    randomNumbers(x) = randomclass.Next(1, 101)
Next x 'increment x

'add together the numbers in the array
For count = 0 To 99 'loop from array e  t 0 to element 9

    'add the number in random  ar  s position count to t
    total = total + rand  b rs(count)
Next count 'incre  ount

Cons  te ne("The total is " & total) 'output the tota
```

2D Arrays

Description of Code

A 2D array has the same principles as a 1D array, but when visualised, there is

An array can be visualised as a table; for example:

Index	0	1	2	3
0	"Red"	"Blue"	"Orange"	"Yellow"
1	"Yellow"	"Red"	"Blue"	"Purple"
2	"Red"	"Yellow"	"Purple"	"Blue"

The array now has two indices. It does not matter which is the x (across) valu value as long as they are used in the same way consistently in your program.

Declaration

Like a standard array or variable, a 2D array needs to be declared.

```
Dim <identifier>(<start number> to <end number>,
<end number>) As <data type>
```

For example, to declare an array named **colours** with five elements that store

```
Dim colours(0 to 4,0 to 2) As String
```

Adding data

Data can be added into a 2D array element:

```
<identifier>(<index>,<index>) = <data>
```

For example, to store "Blue" in element (1,2) of colours:

```
colours(1,2) = "Blue"
```

Accessing data

Data in a specific array element can be accessed:

```
<identifier>(<index>,<index>)
```

For example, to access the data in colours in index (1,2):

```
colours(1,2)
```

Code in Context

1. Store shades of four colours in a list; the first 'column' is shades of red, the
   Ask the user to input a colour, and then output the three shades stored fo

```
'declare an array that is 4 elements by 3 elements
Dim colours(0 To 3, 0 To 2) As String
Dim userColour As String
Dim column As Integer

'store shades of colour in the array
'shades of red
colours(0, 0) = "Cherry"
colours(0, 1) = "Pink"
colours(0, 2) = "Magenta"

'shades of blue
colours(1, 0) = "Powder Blue"
colours(1, 1) = "Aquamarine"
colours(1, 2) = "Navy"

'shades of yellow
colours(2, 0) = "Lemon"
colours(2, 1) = "Gold"
colours(2, 2) = "Cream"

'shades of green
colours(3, 0) = "Avocado"
colours(3, 1) = "Lime"
colours(3, 2) = "Olive"

'output the message
Console.WriteLine("Enter a colour: Red, Blue, Yellow or Gre

'read in input, convert to upper case, store in userColou
userColour = UCase(Console.ReadLine)

If userColour = "RED" Then 'if the user has entered Red
    column = 0 'set the column to 0
ElseIf userColour = "BLUE" Then 'if the user has entered Bl
    column = 1 'set the column to 1
ElseIf userColour = "YELLOW" Then 'if the user has entered
    column = 2 'set the column to 2
Else 'otherwise
    column = 3 'set the column to 3
End If

'output all the shades of the colour entered
For count = 0 To 2 'loop for each 'row' in the array
    'output the data store in colours at element (column, c
    Console.WriteLine(colours(column, count))
Next count 'increment count
```

2. The list stores 10 players, and five scores for each player. Ask the user to
in each round. Ask the user which player and round they would like to vie

```
Dim numbers(0 To 9, 0 To 4) As Integer 'declare a 2D array
Dim round, player As Integer

For y = 0 To 4 'loop through each round
    For x = 0 To 9 'loop through each player

        'output the player and round
        Console.WriteLine("Enter the scores for player " & x +

        'read user input and store in numbers at position (
        numbers(x, y) = Console.ReadLine
    Next x 'increment x
Next y 'increment y

Console.WriteLine("Enter the round number") 'output the mes
round = Console.ReadLine() 'read the input, store it in rou
Console.WriteLine("Enter the player number") 'output the me
player = Console.ReadLine() 'store it in player

'output the player number and round entered, and the player
Console.WriteLine("Player " & player & " scored " & numbers(
& " in round " & round)
```

3. Store random numbers between 0 and 100 in each element of the list nu
'column' they would like the average calcula[illegible] work out the mean avera

```
Dim randomclass As New Random 'declare randomclass as a set

'declare an array of numbers of type integer, with 4 'columns
Dim numbers(3, 0 To 4) As Integer
Dim user[illegible]umn As Integer
Dim average As Single = 0 'initialise average to be 0

'store a random number in each array element
For column = 0 To 3 'loop through each column
    For row = 0 To 4 'loop through each row
        'generate a random number between 0 and 100, store
        '(column, row)
        numbers(column, row) = randomclass.Next(0, 101)
    Next row 'increment row
Next column 'increment column

'ask the user which column they would like to calculate the
Console.WriteLine("Which column would you like the average

'subtract one from the column number to make the index, sto
userColumn = Console.ReadLine() - 1

'work out column mean average
For row = 0 To 4 'loop through each row

    'add the value in numbers as index (userColumn, row) to
    average = average + numbers(userColumn, row)
Next row 'increment row

average = average / 4 'calculate the mean by dividing the t
Console.WriteLine(average) 'output the data in average
```

Code in Context

4. Store the row and column number of each element, in that element. Outp
   as a grid.

```
'declare an array, numbers of type integer, with 10 'column
Dim numbers(0 To 9, 0 To 9) As String

For column = 0 To 9 'loop through each column
    For row = 0 To 9 'loop through each row

        'store the column and row number as a string in num
        'column, row
        numbers(column, row) = Str(column) & Str(row)
    Next row 'increment row
Next column 'increment column

'output each array element with a | between
For column = 0 To 9 'loop through each column
    For row = 0 To 9 'loop through each row

        'output data in that column and row of numbers
        Console.Write(numbers(column, row))
        Console.Write(" |") 'output a pipe character
    Next row
    Console.WriteLine() 'force a new line in the output
Next column
```

Array Tools

Description of Code

There are a number of inbuilt methods that be used on arrays. The tools here c
on a 1D array.

Sort

Arrays have a built-in sort function t o an array into ascending order:

```
System.Array.So (<array identifier>)
```

Reverse

The reverse function reverses the elements in the array:

```
System.Array.Reverse(<array identifier>)
```

Search

A built-in binary search can be called to find out whether a specific data item
sorted in ascending order before a binary search is performed. The function re
occurrence of the data being searched for. -1 is returned if it is not found.

```
System.Array.BinarySearch(<array identifier>,<da
```

Code in Context

1. Input 10 numbers into a list. Sort the l in ascending order and outpu

```
Dim values(0 To 9) As Integer 'declare a 1D array with 10 e

For x  To  'loop from the first element to the last
    va x) = Console.ReadLine() 'read a value from the us
Next x 'increment x

System.Array.Sort(values) 'sort the contents of the array i

'output each element in the array
For x = 0 To 9 'loop from the first element to the last
    Console.Write(values(x) & " ") 'output the value in ele
Next x 'increment x
```

2. Ask the user to enter five colours. Sort the colours into descending order

```
Dim values(0 To 4) As String 'declare a 1D array with 5 ele

Console.WriteLine("Enter 5 colours")
For x = 0 To 4 'loop from  t element to the last
    values(x) = Cons .Re Line() 'read a value from the us
Next x 'increme

System .Sort(values) 'sort the contents of the array i

'output each element in the array
For x = 0 To 4 'loop from the first element to the last
    Console.Write(values(x) & " ") 'output the value in ele
Next x 'increment x
```

3. Ask the user to enter numbers until they do not enter "Y" to continue. Ou
entered, the smallest and largest numbers, and how many times they ente

```
Dim values(0 To 4) As String 'declare a 1D array with 5 ele

Console.WriteLine("Enter 5 colours") 'output the message
For x = 0 To 4 'loop from the first element to the last
    values(x) = Console.Read'L... 'read a value from the us
Next x 'increment x

System...(values) 'sort the contents of the array i
System...Reverse(values) 'reverse order of items in val

'output each element in the array
For x = 0 To 4 'loop from the first element to the last
    Console.Write(values(x) & " ") 'output the value in ele
Next x 'increment x
```

4 Ask the user which colour they would like to remove from a list and delete

```
Dim randomclass As New Random
Dim values(0 To 14) As Integer 'declare a 1D array with 15
Dim userInput As Integer
Dim position As Integer = -1

For x = 0 To 14 'loop from the first elem... to the last

    'generate a random number be...nd 100, store it i
    values(x) = randomclas...e(..., 101)
Next x 'increment x

Consol...t(... "What number would you like to search for
userIn... Console.ReadLine() 'read the user input, store

System.Array.Sort(values) 'sort the array into ascending or

'search the array values for the value in userInput, store
position = System.Array.BinarySearch(values, userInput)
If position <> -1 Then 'if the element is found (the number
    Console.WriteLine("Found it at position " & position)
Else 'otherwise
    Console.WriteLine("Couldn't find it") 'output that it w
End If
```

Reading from a

Description of Code

A text file lets you store data external to the program fi[illegible] is means the data another program, or by the same program when [illegible] running again. If you program, e.g. in an array, it will disappe[illegible] the program stops running.

To read the data from a fi[illegible] you first need to make sure the text file is in a suita the same fol[illegible] th[illegible] xecutable file (.exe) for the program. Then you only nee folders as we[illegible]

The following code opens a file to be read:

```
Dim <identifier> As New System.IO.StreamReader(<
```

The first line of text is read into the program using:

```
<identifier>.ReadLine()
```

For example:

```
Dim fileRead As New System.IO.StreamReader("myDa
console.writeline(fileRead.ReadLine())
```

As soon as you have finished working with a file, you should close it:

```
<identifier>.Close()
```

Reading all t[illegible] file

If you want t[illegible] all the text from a file, you need to continue until you reac function that returns -1 if there is no more data in the file.

For example, the following code will output each line until there are no more

```
While <identifier>.Peek <> -1
    <identifier>.readLine()
End While
```

Code in Context

1. Output each line in the file "data file 1.txt".

```
Dim fileName = "data file 1.txt" 'store the file name of th
Dim readFile As New System.IO.StreamReader(fileName) 'open

While readFile.Peek <> -1 'while there is still another lin
    Console.WriteLine(readFile.ReadLine()) 'read line from
End While
readFile.Close() 'close the file
```

2. Ask the user which line in "data file 2.txt" they want to output, then output

```
Dim fileName = "data file 2.txt" 'store the file name of th
Dim readFile As New System.IO.StreamReader(fileName) 'open

Dim userInput, counter As Integer
Console.WriteLine("Which line would you like to read?") 'ou
userInput = Console.ReadLine() 'read the user input, store

counter = 1 'store 1 in counter

While readFile.Peek <> -1 'while there is still another lin
    'if value in counter is the same as the value in userIn
    If counter = userInput Then
        'read the line from the file and output it
        Console.WriteLine(readFile.ReadLine())
    End If
    counter = counter + 1 'increment counter
End While

readFile.Close() 'close the file
```

3. Read the data from a file into an array.

```
Dim fileName = "data file 2.txt" 'store the file name of th
Dim readFile As New System.IO.StreamReader(fileName) 'open

'declare array with 100 elements to store data from the fil
Dim fileData(0 To 99) As String

Dim counter As Integer = 0 'store 0 in counter

While readFile.Peek <> -1 'while there is still another lin
    'read the line from the file and store it in the array
    fileData(counter) = readFile.ReadLine()
    counter = counter + 1 'increment counter
End While

'output number of lines read
Console.WriteLine("There were " & counter & "lines of data

readFile.Close() 'close the file
```

4. Read the data in a file into a list. Output the three shades of a colour the

```
Dim fileName = "data file 3.txt" 'store the file name of th
Dim readFile As New System.IO.StreamReader(fileName) 'open
Dim fileData(0 To 3, 0 To 2) As String 'declare a 2D array
Dim userInput As String

Dim counter As Integer = 0 '  0 as counter

'read the data i  array
While re Fi  <> -1 'while there is still another lin

    're  he first three lines of data into the array elem
    For x = 0 To 2 'loop from first element in first dimens
        fileData(counter, x) = readFile.ReadLine() 'read li
    Next x 'increment x

    counter = counter + 1 'increment counter
End While

'read which colour the user wants to output
Console.WriteLine("Which colour would you like the shades o
blue or green?")
userInput = UCase(Console.ReadLine())

For x = 0 To 3 'loop through the first elements
    If fileData(x, 0) = userInput Then 'if  ta in the elem
        For y = 0 To 2 'loop through t e second dimension
            Console.WriteLine(f eD t (x, y)) 'output that
        Next
    End If
Next x

readFi  se() 'close the file
```

Appending to a

Description of Code

"Appending' means 'adding to the end of'. Appending to a file means the data
end of the data that is already in the file, instead of o ng it.

The following code opens a file to be ap d to:

```
Dim <identifier> As New System.IO.StreamWriter(<
```

The first line t is read into the program using:

```
<identifier>.WriteLine(<data>)
```

For example:

```
Dim fileWrite As New System.IO.StreamWriter("myD
fileWrite.WriteLine("Bob")
```

As soon as you have finished working with a file, you should close it:

```
<identifier>.Close()
```

Code in Context

1. Write the number 11 to the end of the text file.

```
Dim f e "data file 1.txt" 'store the file name of th

'open file to write in append mode
Dim fileAppend As New System.IO.StreamWriter(fileName, True)

fileAppend.WriteLine(11) 'append 11 to the end of the text
Console.WriteLine("Data written") 'output the message

fileAppend.Close() 'close the file
```

2. Add the strings "White" and "Silver" to the end of the text file.

```
Dim fileName = "data file 2.txt" 'store the file name of th

'open file to write in append mode
Dim fileAppend As New System.IO.StreamWriter(fileName, True)

fileAppend.WriteLine("White") 'append "White" to the end of
fileAppend.WriteLine("Silver") 'append "Silver" to the end
Console.WriteLine("Data written") 'output the message

fileAppend.Close() 'close the file
```

3. Ask the user to input 10 numbers and append these to the end of the text

```
Dim fileName = "data file 1.txt" 'store the file name of th

'open file to write in append mode
Dim fileAppend As New System.IO.StreamWriter(fileName, True)

Console.WriteLine("Enter 10 numbers") 'ask the user to inpu
For x = 0 To 9 'loop 10 times

    'append user data to the end of the text file
    fileAppend.WriteLine(Console.ReadLine())
Next x 'increment x

Console.WriteLine("Data written") 'output the message

fileAppend.Close() 'close the file
```

4. Input a colour and two shades of that colour. Append them to a text file.

```
Dim fileName = "data file 3.txt" 'store the file name of th

'open file to write in append mode
Dim fileAppend As New System.IO.StreamWriter(fileName, True)
Dim userInput As String

'ask the user to input a colour
Console.WriteLine("Enter a colour of light (e.g. red, yellow, b
userInput = Console.ReadLine() 'read the colour and store i

'append the colour the user entered to the end of the file
fileAppend.WriteLine(userInput)

For x = 1 To 2 'loop twice

    'ask the user to input a shade of the colour they enter
    Console.WriteLine("Input a shade of " & userInput)
    fileAppend.WriteLine(Console.ReadLine()) 'append the sh
Next x

Console.WriteLine("Data written") 'output the message

fileAppend.Close() 'close the file
```

Overwriting a I

Description of Code

Overwriting means that the data in the file will be deleted and then you can a
want to write.

The following code opens a file to be written:

```
Dim <identifier> As New System.IO.StreamWrite
```

The first line that is read into the program using:

```
<identifier>.WriteLine(<data>)
```

For example:

```
Dim fileWrite As New System.IO.StreamWriter("
fileWrite.WriteLine("Bob")
```

As soon as you have finished working with a file, you should close it:

```
<identifier>.Close()
```

If the file does not exist in the location given, it will be created for you.

Code in Context

1. Write "Hello World" to a text file

```
Dim filename As String = "file1.txt" 'store the file name a

'open file filename to write data to
Dim newFile As New System.IO.StreamWriter(filename)

newFile.WriteLine("Hello World") 'write the string to the f
Console.WriteLine("Data written") 'output the message

newFile.Close() 'close the file
```

2. Write a user's first name and surname to a text file.

```
Dim filename As String = "userData.txt" 'store the file nam
Dim firstName, surname As String

'read the user's first name and store in firstName
Console.WriteLine("Enter your first name")
firstName = Console.ReadLine()

'read the user's surname and store in surname
Console.WriteLine("Enter your surname")
surname = Console.ReadLine()

Dim newFile As New System.IO.StreamWriter(filename) 'open f
newFile.WriteLine(firstName) 'write the data in firstName o
newFile.WriteLine(surname) 'write the data in surname on th

newFile.Close() 'close the file
```

3. Ask the user how many numbers they want to enter. Let them enter this n
them to a text file separated by commas.

```
Dim filename As String = "numbers.txt" 'store the file name
Dim userInput As Integer

'read how many numbers the user wants to enter
Console.WriteLine("How many [illegible]s would you like to enter
userInput = Console.[illegible]

Dim ne[illegible] = [illegible]System.IO.StreamWriter(filename) 'open f

For co[illegible] = 1 To userInput 'loop from 1 to the number the
    Console.WriteLine("Enter number " & counter) 'ask the u
    newFile.WriteLine(Console.ReadLine()) 'write the number
Next counter 'increment counter

Console.WriteLine("Data written")

newFile.Close() 'close the file
```

4. Store the data in an array in a text file.

```
Dim filename As String = "arrayData.txt" 'store the file na
Dim theData(0 To 1, 0 To 2) As String 'declare a string wit

'store the colours in the array
theData(0, 0) = "Red"
theData(0, 1) = "Crimson"
theData(0, 2) = "P[illegible]
theData(1, 0) = [illegible]e
theDa[illegible]1, [illegible] "Navy"
theDat[illegible]2) = "Azure"

Dim newFile As New System.IO.StreamWriter(filename) 'open f

For column = 0 To 1 'loop through the first dimension of th
    For row = 0 To 2 'loop through the second dimension of
        newFile.WriteLine(theData(column, row)) 'write the
    Next row 'increment row
Next column 'increment column

Console.WriteLine("Data written")

newFile.Close() 'close the file
```

Subroutines: Proc

Description of Code

A subroutine is an independent piece of code, with its own identifier, that can
other places within the program.

A *procedure* is a specific type of subroutine that does not return any values to
called it. You can send data (parameters) to the procedure to be used within it.

Declaring a procedure

A procedure is declared using the code:

```
Sub <identifier> (<parameters>)
     <subroutine code>
End Sub
```

Example without parameters:

```
Sub myProcedure()
     Console.writeline("Hello World")
End Sub
```

Example with parameters:

```
Sub myProcedure(ByVal x as Integer)
     Console.writeline(x)
End Sub
```

Declaring a procedure

A procedure is called by using its name (identifier).

Example without parameters:

```
myProcedure()
```

Example with parameters:

```
myProcedure(10)
```

Code in Context

1. Call a function to ask the user to enter their name. Then call a function t

```
Sub Main()

    firstOutputs() 'call the procedure firstOutputs
    secondOutputs() 'call the procedure secondOutputs

End Sub

Sub firstOutputs() 'declare the procedure firstOutputs
    Dim name As String
    Console.WriteLine("What is your name?") 'ask the user t
    name = Console.ReadLine() 'read the input and store in
    Console.WriteLine("Hello " & name) 'Output hello and th
End Sub 'return control to the program that called it

Sub secondOutputs() 'declare the procedure secondOutputs
    Dim age As Integer
    Console.WriteLine("How old are you in years?") 'ask the
    age = Console.ReadLine() 'read the input and store it i

    Console.WriteLine("You are " & age & " years old") 'out
End Sub 'return control to the program that called it
```

2. Ask the user to input numbers until they say "No". Output if each number
or equal to) 10.

```
Sub Main()
    Dim userInput As String
    Dim theNumber As Integer

    Do 'start loop
        Console.WriteLine("Enter a number") 'output the str
        theNumber = Console.ReadLine() 'read the user input

        'call the procedure outputValue, send the data in t
        'as a parameter
        outputValue(theNumber)

        Console.WriteLine("Again?") 'output the string
        userInput = Console.ReadLine() 'read the user input

    'if userInput as capital letters is equal to "NO", retu
    Loop Until (UCase(userInput) = "NO")

    Console.ReadLine()

End Sub

'declare a procedure outputValue that has the parameter n
Sub outputValue(ByVal number As Integer)

    If number > 10 Then 'if the value in number is greater
        Console.WriteLine("Greater than 10") 'output the st
    ElseIf number < 10 Then 'if not, but the value in numbe
        Console.WriteLine("Less than 10") 'output the strin
    Else 'if neither if is true
        Console.WriteLine("It is 10") 'output the string
    End If

End Sub
```

3. A short text-based game: the user chooses from options and the story co

```
Sub Main()
    Dim doorChoice As String

    'output the story
    Console.WriteLine("Welcome to the fi  oom")
    Console.WriteLine("You are in a  om and have the choic
    Console.WriteLine("Would  ou like to go through the left
    doorChoice = Conso  e.ReadLine() 'read the user input an

    If UCase(do  Choice) = "LEFT" Then 'if the user entered
        Ft  r() 'call the procedure leftDoor
    El  therwise
        ightDoor() 'call the procedure rightDoor
    End If

    Console.ReadLine()

End Sub

Sub leftDoor() 'declare a procedure called leftDoor
    Dim userInput As String

    'output the story
    Console.WriteLine("You have gone through the left door")
    Console.WriteLine("You are in a room with no doors, but
    brick on the floor")
    Console.WriteLine("Would you like to go back, or use th
    Console.WriteLine("Enter Back or Window")
    userInput = Console.ReadLine() 're  e user input and

    If UCase(userInput) = "BACK" Th  if the value in user
        Main() 'call th  ocedure
    Else 'otherwis
        brick  () 'call the brickWindow procedure
    E
End Su

Sub rightDoor() 'declare a procedure called rightDoor
    Dim userInput As String

    'output the story
    Console.WriteLine("There is a door in front of you and
    Console.WriteLine("Would you like to go through the door
    Console.WriteLine("Enter Door, Trapdoor or Back")
    userInput = Console.ReadLine() 'read the user input and

    If UCase(userInput) = "BACK" Then 'if userInput is equa
        Main() 'call the Main procedure
    ElseIf UCase(userInput) = "DOOR" Then 'if not, but it i
        rightDoor() 'call the rightDoor procedure (they sta
    Else 'if neither if is true
        trapdoor() 'call the proced  e t  pdoor
    End If
End Sub

Sub brickWindow()  eclare a procedure called brickWindow
    C  .  iteLine("Well done - you escaped") 'output th
    C  .ReadLine()
End Sub

Sub trapdoor() 'declare a procedure called trapdoor
    Console.WriteLine("Oh dear, you fell to your death.  Game
    Console.ReadLine()
End Sub
```

Code in Context

4. Read two numbers from the user; output if the first number is (or is not) g

```
Sub Main()
    Dim number1 As Integer
    Dim number2 As Integer
    number1 = Console.ReadLine() 'read the input and store
    number2 = Console.ReadLine() 'read the input and store

    If number1 > number2 Then 'if the value in num1 is grea
        'call procedure method1, send number1 and number2 a
        method1(number1, number2)
    Else 'otherwise
        'call procedure method2, send number1 and number2 a
        method2(number1, number2)
    End If

    Console.ReadLine()
End Sub

'declare a procedure called method 1, taking two parameters
Sub method1(num1 As Integer, num2 As Integer)
    Console.WriteLine(num1 & " is greater than " & num2) 'o
End Sub

'declare a procedure called method 2, taking two parameters
Sub method2(num1 As Integer, num2 As Integ
    Console.WriteLine(num1 & " is not greater than " & num2)
End Sub
```

Subroutines: Func

Description of Code

A subroutine is an independent piece of code, with its own identifier, that can
places within the program.

A *function* is a specific type of subroutin which returns a value to the program t
You can send data (parameters) to run the procedure to be used within it.

A value must be returned. This can be done using the Return command word,
assigning a v to the function name.

Declaring a function

A function is declared using the code:

```
Function <identifier> (<parameters>) As <data ty
     <subroutine code>
     Return <identifier>
End Function
```

Example without parameters:

```
Function myFunction() As String
     <subroutine code>
     Return "Hello"
End Function
```

Example wit m:

```
Function myFunction(ByVal x as Integer) As Integ
     <subroutine code>
     myFunction = x
End Function
```

Calling a function

A function is called using its identifier (with any required parameters in bracke
replaced with the value returned, so something needs to be done with this, e.g.
variable or used in a comparison.

Example without parameters:

```
myString = myFunction()
```

Example with parameters:

```
Console.WriteLine(myFunction(10))
```

Code in Context

1. A function adds together two numbers and returns the result.

```
Sub Main()
    Dim result As Integer

    'call calculateResult with 10 and 20 as parameters, sto
    'in result
    result = calculateResult(10, 20
    Console.WriteLine(result) 'output the value in result
    Console.ReadLine()
End Sub

'declare a function named calculateResult.
'take two parameters, num1 and num2
'return an integer value
Function calculateResult(num1 As Integer, num2 As Integer) A
    calculateResult = num1 + num2 'return the result of num
End Function
```

2. Use a function to check that a value input by the user is valid.

```
Sub Main()
    Dim userInput As Integer
    Console.WriteLine("Enter a number between 0 and 100") '
    userInput = Console.ReadLine() 'read the user input and

    'call the function validateInput with the value in user
    If validateInput(userInput) = True Then 'if the value r

        'multiply the value in userInput by 10 and store in
        userInput = userInput * 10

        'output the new value of userInput
        Console.WriteLine("Thanks for entering a valid numb
        Console.WriteLine(userInput)
    Else 'if the value returned is not true

        'output that it is invalid
        Console.WriteLine(userInput & " is not valid; it's
        userInput = 0 'store 0 in userInput
    End If

    Console.ReadLine()
End Sub

'declare a function named validateInput, that takes one int
'and returns a Boolean
Function validateInput(theNumber As Integer) As Boolean

    'if the value in the parameter is greater than or equal
    'equal to 100
    If theNumber >= 0 And theNumber <= 100 Then
        Return True 'return the Boolean True
    Else 'otherwise
        Return False 'return the Boolean False
    End If
End Function
```

3. Three numbers are read from the user. The function returns the integer d
divided by the third number, plus the second number.

```
Sub Main()
    Dim firstNum, secondNum As Integer

    Console.WriteLine("Enter the first number") 'output the
    firstNum = Console.ReadLine() 'read the user input and
    Console.WriteLine("Enter the second number") 'output th
    secondNum = Console.ReadLine() 'read the user input and

    'if value in firstNum is greater than or equal to t
    If firstNum >= secondNum Then

        'call function calculateValue with firstNum and sec
        'output the value returned from the function call
        Console.WriteLine(calculateValue(firstNum, secondNu
    Else
        'call function calculateValue with secondNum and fi
        'output the value returned from the function call
        Console.WriteLine(calculateValue(secondNum, firstNu
    End If

    Console.ReadLine()
End Sub

'declare a function called calculateValue that takes two in
'parameters and returns an integer number
Function calculateValue(number1 As Integer, number2 As Inte
    Dim thirdNum As Integer

    'output the first parameter value
    Console.WriteLine(number1 & " is the largest number")
    Console.WriteLine("Enter a new number")
    thirdNum = Console.ReadLine() 'read the number from the u

    'find the integer division of number1 DIV thirdNum, then
    'store the result in calculateValue to be returned to t
    calculateValue = (number1 \ thirdNum) + number2

End Function
```

4. Take as input two numbers and a calculation. Call a specific function bas
the result of the calculation.

```
Sub Main()
    Dim firstNum, secondNum As Integer
    Dim symbol As String

    Console.WriteLine("Enter the fi st n ber") 'output the
    firstNum = Console.ReadL  'read the user input and
    Console.WriteLine(" e second number") 'output th
    secondNum = C  ReadLine() 'read the user input and

    C  .W  Line("Would you like to +, -, / or *? Enter
    sy  = Console.ReadLine() 'read the user input and st

    'jump to the Case statement based on the value in symbo
    Select Case symbol
        Case "+"
            'call function addNumbers, with firstNum and se
            Console.WriteLine(addNumbers(firstNum, secondNu
        Case "-"
            'call function subtractNumbers, with firstNum a
            Console.WriteLine(subtractNumbers(firstNum, second
        Case "/"
            'call function divideNumbers, with firstNum and
            Console.WriteLine(divideNumbers(firstNum, secondN
        Case "*" '
            'call function multiplyNumbers, with firstNum a
            Console.WriteLine(multiplyNumbers(firstNum, second
    End Select

    Console.ReadLine()
End Sub

'declare a func  called addNumbers which it takes two pa
'retur  s  integer
Functi dNumbers(num1 As Integer, num2 As Integer) As Si
    ad  bers = num1 + num2 'return the result of num1 + n
End Function

'declare a function called subtractNumbers
'it takes two parameters as integers and returns a single i
Function subtractNumbers(num1 As Integer, num2 As Integer)
    If num1 > num2 Then 'if value in num1 is greater than
        subtractNumbers = num1 - num2 'return the result of
    Else 'otherwise
        subtractNumbers = num2 - num1 'return the result of
    End If
End Function

'declare a function called multiplyNumbers
'it takes two parameters as integers and  rns a single i
Function multiplyNumbers(num1 As In er  um2 As Integer)
    multiplyNumbers = num1 *  'return the result of num
End Function

'declare a func  called divideNumbers
'it t  w  meters as integers and returns a single i
Functi ideNumbers(num1 As Integer, num2 As Integer) As
    If  m1 > num2 Then 'if value in num1 is greater than va
        divideNumbers = num1 / num2 'return the result of n
    Else 'otherwise
        divideNumbers = num2 / num1 'return the result of n
    End If
End Function
```

Subroutines: Para

Description of Code

A parameter is a variable whose value is sent to the subroutine by the program

A subroutine declares the parameters it needs when it is defined. A subroutine number of parameters.

Procedure parameter

```
Sub <identifier> (<parameters>)
    <subroutine code>
End Sub
```

For example:

```
Sub myProcedure(ByVal x as Integer)
    Console.writeline(x)
End Sub
```

Function parameter

A procedure is declared using the code:

```
Function <identifier> (<parameters>) As <data ty
    <subroutine code>
    Return <identifier>
End Function
```

Example with parameters:

```
Function myFunction(ByVal x as Integer) As Integ
    <subroutine code>
    myFunction = x
End Function
```

Sending parameters

If a subroutine is declared with a parameter, then a value must be sent when it or a variable that contains the data.

For example:

```
myProcedure(10)   or   myProcedure(variableOne)
```

By Value or by Reference

A parameter can be sent by value or by reference. If neither is chosen, then by

- **ByVal** – the data in the variable is sent; the variable's value is not ch
- **ByRef** – the location of the variable is sent; the variable's value is ac

Code in Context

1. When the parameter is multiplied by 10, the value in the original variable

```
Sub Main()
    Dim number As Integer = 10 'store 10 in the variable number

    calculateNumber(number) 'call the procedure, send the variable n

    'output the value in number
    Console.WriteLine("After the procedure, the original number is "
    Console.ReadLine()
End Sub

'declare a procedure called calculateNumber which takes a parameter n
'(it takes and edits the value in the variable, not the variable)
Sub calculateNumber(ByVal number As Integer)
    number = number * 10 'multiply the value in number by 10, store
    Console.WriteLine("The procedure number is " & number) 'output t
End Sub
```

2. When the parameter is multiplied by 10, the value in the original variable

```
Sub Main()
    Dim number As Integer = 10 'store 10 in the variable number

    calculateNumber(number) 'call the procedure, send the variable n

    'output the value in number
    Console.WriteLine("After the procedure, the original number is "
    Console.ReadLine()
End Sub

'declare a procedure named calculateNumber which takes a parameter n
' (it takes and edits the value in the original variable)
Sub calculateNumber(ByRef number As Integer)
    number = number * 10 'multiply the value in number by 10, store
    Console.WriteLine("The procedure number is " & number) 'output t
End Sub
```

3. The program takes a number from the user then loops 10 times, sending the user has entered to a procedure. The procedure adds the numbers to

```
Sub Main()
    Dim firstNum As Integer
    Dim secondNum As Integer

    firstNum = Console.ReadLine() 'read the input store in firstNum

    For x = 0 To 9 'loop 10 times
        secondNum = Console.ReadLine() 'read the input store in seco

        'call the procedure outputMessages. The value in firstNum is
        'secondNum is the second.
        outputMessages(firstNum, secondNum)
    Next x 'increment x

    Console.ReadLine()
End Sub

'declare a procedure named outputMessages which takes two parameters,
'num1, the second be
Sub output
    Conso
    Conso
    Console.WriteLine("Together they make " & (num1 + num2)) 'output th
End Sub
```

Searching

Description of Code

If data is stored in a structure such as an array, you may need to search it to fi
exists, or to find the location of an item.

There are many method of searching; some ... e efficient in specific scena

A *linear* search goes through ... item in a list, one at a time, from the first el
until it finds what it ... for or reaches the end of the list.

A *binary* sear... ds a list of data to be in order. It then takes the middle ele
item it is looking for. If the middle element is smaller than the item it is looki
search just on the right-hand side of the list (all the elements greater than the
it repeats it with all those elements that are smaller. This is repeated until the
finds the item it is looking for.

A range of searching methods are given in the examples below.

Code in Context

1. Search an array to find out whether an item exists, and then output if it is

```
Dim randomClass As New Random 'declare the random class

Sub Main()
    Dim itemArray(0 To 9) As Integer 'declare array itemArra

    'generate 10 random numbers and store them in the array
    For count = 0 To 9 'loop from the first array element t
        'generate a random number between 1 and 100 and sto
        itemArray(count) = randomClass.Next(1, 101)
    Next count 'increment count

    Dim userInput As Integer
    Console.WriteLine("Enter the number you want to find")
    userInput = Console.ReadLine() 'read the user input and

    'set flag to be False. If false, it means the item has
    Dim flag As Boolean = False

    'loop through each element in the array and check if it
    For count2 = 0 To 9 'loop from the first element to the

        'check if the current array item is equal to userIn
        If itemArray(count2) = userInput Th
            flag = True 'if true, set flag to be True
        End If
    Next

    If flag = True Then 'if the value in flag is True
        Console.WriteLine("It was in the array") 'output th
    Else 'if the value in flag is False
        Console.WriteLine("It was not in the array") 'outpu
    End If

    Console.ReadLine()
End Sub
```

2. Check whether an item exists in a 1D array. If it does, output all the posit
failure message if it does not exist.

```
Dim randomClass As New Random 'declare the random class

Sub Main()
    Dim itemArray(0 To 9) As Integer 'decl  array called it

    'generate 10 random numbers and st  them in the array
    For count = 0 To 9 'loop from the first array element t

        'generate a random number between 1 and 100 and sto
        item  count) = randomClass.Next(1, 11)
    N  ur  'increment count

    Dim userInput As Integer
    Console.WriteLine("Enter the number you want to find")
    userInput = Console.ReadLine() 'read the user input and

    'set flag to be False. If false, it means the item has
    Dim flag As Boolean = False

    'loop through each element in the array and check if it
    For count2 = 0 To 9 'loop from the first element to the
        If itemArray(count2) = userInput Then 'check if array
            flag = True 'if true, set flag to be True
            Console.WriteLine("The item is at position " & cou
        End If
    Next

    If flag = False Then 'if the item was  found
        Console.WriteLine("It was n  in the array") 'outpu
    End If

    Console.ReadLine()
End Sub
```

3. Search a array for the last location of an item.

```
Dim randomClass As New Random 'declare the random class

Sub Main()
    Dim itemArray(0 To 9) As Integer 'declare array called i

    'generate 10 random numbers and store them in the array
    For count = 0 To 9 'loop from the first array element t

        'generate a random number between 1 and 100 and sto
        itemArray(count) = randomClass.Next(1, 11)
    Next count 'increment count

    Dim userInput As Integer
    Console.WriteLine("Enter the nu be  ou want to find")
    userInput = Console.Re  'read the user input and

    Array.Sort(i  ' sort the array into ascending or

    '  array itemArray for the item userInput.  Bi
    'l  ndex as an integer, or a minus number if it does
    Console.WriteLine(Array.BinarySearch(itemArray, userInput))

    Console.ReadLine()
End Sub
```

4. Search a 2D array for the location of an item. Output the location if found, if it is not found.

```
Dim randomClass As New Random 'declare the random class

Sub Main()

    'declare a 2D array with 10 spaces by 20 spaces, named
    Dim itemArray(0 To 9, 0 To 19) As Integer

    'generate random numbers and store them in the array
    For column = 0 To 9 'loop through each column in the 2D
        For row = 0 To 19 'loop through each row in the 2D

            'generate a random number between 1 and 10 and
            itemArray(column, row) = randomClass.Next(1, 11)
        Next row 'increment row
    Next column 'increment count

    Dim userInput As Integer
    Console.WriteLine("Enter the number you want to find")
    userInput = Console.ReadLine() 'read the user input and

    'set flag to be False. If false, it means the item has
    Dim flag As Boolean = False

    'loop through each element in the array and check if it
    For column = 0 To 9 'loop from the first dimension
        For row = 0 To 19

            'check if the current array item is equal to us
            If itemArray(column, row) = userInput Then
                flag = True 'if true, set flag to be True

                'output the position of the item
                Console.WriteLine("The item is at position
            End If
        Next row
    Next column

    If flag = False Then 'if the item was not found
        Console.WriteLine("It was not in the array") 'outpu
    End If

    Console.ReadLine()
End Sub
```

Sort

Description of Code

If data is stored in a structure such as an array, you may need to sort it in descending order.

There are set functions that can do this, or you can write your own sorting algo It makes more sense to use the in-built functions in VB.NET.

There are a range of sorting methods; some are more efficient than others dep

A *bubble* sort compares the first and second items in a list of values. If they ar them. It then repeats this with the second and third items, then the third and f end of the list, it checks whether there have been any swaps. If there have, it g and starts again. If there haven't, then the list is sorted.

A *merge* sort splits each element into its own list. It then merges pairs of indiv It repeats this with pairs of ordered lists, and merges them into a larger, ordere the lists have been merged into one.

Code in Context

1. Sort the items in an array into ascending order using the sort function.

```
Dim randomClass As New Random 'declare a random class

Sub Main()
    Dim itemArray(0 To 9) As Integer 'declare a 1D array with

    Console.WriteLine("Before")
    'generate 10 random numbers nad store them in the array
    For count = 0 To 9 'loop from the first array element t

        'generate a random number between 1 and 100 and sto
        itemArray(count) = randomClass.Next(1, 11)
        Console.WriteLine(itemArray(count)) 'output the arr
    Next count 'increment count

    Array.Sort(itemArray) 'sort the array into ascending or

    Console.WriteLine("After")
    For count = 0 To 9 'loop through each element of the ar
        Console.WriteLine(itemArray(count)) 'output the arr
    Next

    Console.ReadLine()
End Sub
```

2. Sort the items in an array into descending order using the sort and reverse

```
Dim randomClass As New Random 'declare the random class

Sub Main()

    'declare a 1D array with 10 [illegible] named itemArray
    Dim itemArray(0 To 9) As [illegible]

    Console.Wri[illegible]ne("Before")

    'generate 10 random numbers and store them in the a
    For count = 0 To 9 'loop from the first array eleme

        'generate a random number between 1 and 100 and
        itemArray(count) = randomClass.Next(1, 11)
        Console.WriteLine(itemArray(count)) 'output the
    Next count 'increment count

    Array.Sort(itemArray) 'sort the array into ascendin

    'reverse the order of the array so it is in descendi
    Array.Reverse(itemArray)

    Console.WriteLine("After")
    For count = 0 To 9 'loop through each element of th
        Console.WriteLine(itemArray(count)) 'output the
    Next

    Console.ReadLine()
End Sub
```

3. Sort a 2D array into ascending order by the first element in the array.

```
Dim randomClass As New Random 'declare the random class

Sub Main()

    'declare a 2D array with 10 spaces by 2 spaces, named i
    Dim itemArray(0 To 9, 0 To 1) As Integer

    'generate 10 random numbers and store them in the array
    For column = 0 To 9 'loop through each column in the 2D
        For row = 0 To 1 'loop through each row in the 2D a

            'generate a random number between 1 and 9 and s
            itemArray(column, row) = randomClass.Next(1, 10)
        Next row 'increment row
    Next column 'increment count

    'output the 2D array before sorting
    Console.WriteLine("Before")
    For x = 0 To 1 'loop for each row

        'output each item in the first index
        Console.WriteLine(itemArray(0, x) & " " & itemArray(
        itemArray(2, x) & " " & itemArray(3, x) & " " & item
        itemArray(5, x) & " " & itemArray(6, x) & " " & item
        itemArray(8, x) & " " & itemArray(9, x))
    Next x

    'bubble sort the array by the first index only
    Dim swap As Boolean = True 'true if a swap is made
    Dim temp1 As Integer = 0
    Dim temp2 As Integer = 0
    While swap = True 'if a swap was made in the last cycle
        swap = False 'set swap to be false

        For count = 0 To 8 'loop 9 times - through each ele

            'if the current item is greater than the next i
            If itemArray(count, 0) > itemArray(count + 1, 0)

                'store the current element in temporary var
                temp1 = itemArray(count, 0)
                temp2 = itemArray(count, 1)

                'replace the current element with the next
                itemArray(count, 0) = itemArray(count + 1, 0
                itemArray(count, 1) = itemArray(count + 1, 1

                'replace the next element with temporary va
                itemArray(count + 1, 0) = temp1
                itemArray(count + 1, 1) = temp2

                'set swap to be True because a swap has bee
                swap = True
            End If
        Next count

    End While

    'output the contents of the 2D array after they have be
    Console.WriteLine("After")
    For x = 0 To 1
        Console.WriteLine(itemArray(0, x) & " " & itemArray(
        itemArray(2, x) & " " & itemArray(3, x) & " " & item
        itemArray(5, x) & " " & itemArray(6, x) & " " & item
        itemArray(8, x) & " " & itemArray(9, x))
    Next x

    Console.ReadLine()
End Sub
```

Random Number Gene

Description of Code

You can generate a random number between any values; this can be a whole n
number. The random function needs to be declared first using the following c

```
Dim <variable name> As New Random()
```

If random numbers need to be generated throughout the program, this can be
subroutines) only when required within a subroutine.

Random numbers are then generated between bounds using the Next functio
number.

```
<variable name>.Next(<lower bound>,<upper
```

For example, the following code will generate a random number between 1 and 1

```
Dim randomNumbers As New Random()
Console.Writeline(randomNumbers.Next(1,11)
```

Code in Context

1. Generate 100 random numbers between 1 and 100.

```
Dim randomNums As New Random() 'declare the random class

Sub Main()
    For x = 0 To 99 'loop 100 times from 0 to 99

        'generate a random number between 1 and 100
        Console.WriteLine(randomNums.Next(1, 101)) 'output

    Next x 'increment x

    Console.ReadLine()
End Sub
```

2. Ask the user how many numbers to generate and what values to generate random numbers, and then output each one and the total of all the gener

```
Dim randomNumbers As New Random() 'declare the random class
Sub Main()

    Dim userInput As Integer
    Console.WriteLine("How many numbers do you want to gene
    userInput = Console.ReadLine() 'read the user input and

    Dim lowest As Integer
    Console.WriteLine("What is the smallest number you want ge
    lowest = Console.ReadLine() 'read the user input and st

    Dim highest As Integer
    Console.WriteLine("What is the largest number you want ge
    highest = Console.ReadLine() 'read the user input and s

    Dim numGenerated, total As Integer

    For count = 1 To userInput 'loop from 1 to the number t

        'generate a random number between the bounds entere
        numGenerated = randomNumbers.Next(lowest, highest +
        Console.WriteLine(numGenerated) 'output the value i

        'add the value in numGenerated to total, store in to
        total = total + numGenerated
    Next count 'increment count

    'output text and value of total
    Console.WriteLine("The numbers all added up to " & total)

    Console.ReadLine()
End Sub
```

3. Generate 100 random numbers between 0 and 1 with up to two decimal p

```
Dim randomNumbers As New Random() 'declare the random class

Sub Main()
    For x = 0 To 99 'loop 100 times

        'generate a random number between 0 and 10, divide
        Console.WriteLine(randomNumbers.Next(0, 11) / 10) 'ou

    Next x 'increment x

    Console.ReadLine()
End Sub
```

Records

Description of Code

In VB.NET you can use a record to store multiple pieces of data with different d

Creating a record structure

Declare a record using the following cod (it can be local or global dependi needs to be used):

```
Public Structure <identifier>
    Dim <identifier> As <data type>
    ...
End Structure
```

For example:

```
Public Structure horses
    Dim name As String
    Dim yearOfBirth As Integer
    Dim colour As String
    Dim height As Integer
End Structure
```

Creating an instance of a record

Once you have declared a structure, this is now a useable data type. So you ca type – your structure name.

```
Dim <identifier> As <record structure name>
```

For example:

```
Dim horseBob As horses
```

Adding data to a record

You can add data to each of the variables within your structure:

```
<identifier>.<record variable identifier> = <dat
```

For example:

```
horseBob.name = Bob
```

Getting data from a record

You can get data from a record by using its variable:

```
<identifier> <record variable identifier>
```

For example:

```
Console.Writeline(horseBob.name)
```

1. Stores the name, year of birth, colour and height of a horse. Outputs all o

```
Public Structure horses 'declare a structure called horses
    'horses has the following variables
    Dim name As String
    Dim yearOfBirth As Integer
    Dim colour As String
    Dim height As Single
End Structure

Sub Main()
    Dim horse1 As horses 'declare a variable horse1 of type horse
    horse1.name = "Bob" 'assign Bob to the name variable of horse
    horse1.yearOfBirth = 2012 'assign 2012 to the yearOfBirth var
    horse1.colour = "Grey" 'assign Grey to the colour variable of
    horse1.height = "14.4" 'assign 14.4 to the height variable of

    'output the data about horse1 in a sentence
    Console.WriteLine("The " & horse1.colour & " horse, " & horse
    horse1.yearOfBirth & " and is " & horse1.height & " hands hig

    Console.ReadLine()
End Sub
```

2. Lets the user enter the favourite colour, age and gender for five people. S
records. Each person has all three pieces of information stored in an arra
average age, and outputs the gender and favourite colour for each person

```
Public Structure peopleInfo 'declare a structure called peopleInf
    'peopleInfo has the following variables
    Dim favColour As String
    Dim age As Integer
    Dim gender As String
End Structure

Sub Main()
    'd  a  array named thePeople, with 5 spaces, of type peo
    Dim  eople(0 To 4) As peopleInfo

    'ask the user to enter the three pieces of information for fi
    For count = 0 To 4
        Console.WriteLine("Enter person number " & count + 1 & "'

        'read the user input, store as the favColour of the curre
        thePeople(count).favColour = Console.ReadLine

        Console.WriteLine("Enter person number " & count + 1 & "'s

        'read the user input, store as age of the current array e
        thePeople(count).age = Console.ReadLine

        Console.WriteLine("Enter person number " & count + 1 & "'

        'read the user input, store as gender of the current arra
        thePeople(count).gender = Console.ReadLine
    Next count

    'calculate the total age of all arr  ments
    Dim totalAge As Integer = 0
    For i = 0 To 4
        totalAge = tot  ge  hePeople(i).age 'add age of curren
    Next i
    Co  Wr  e("The average age of people entered is " & t
    'ou  the gender and colour for each array element
    For  = 0 To 4
        Console.WriteLine(thePeople(i).gender & " and colour " &
    Next i

    Console.ReadLine()
End Sub
```