

Dummy title

Table of Contents

Chapter 1	Introduction	3
1.1	Programming languages	6
Chapter 2	Console Applications	9
2.1	Your first VB.NET program	10
2.2	The Console object	13
2.3	Basic calculations in Visual Basic	17
2.4	Variables in Visual Basic	19
2.5	Writing console applications	23
2.6	Coding style	25
2.7	Exercise	26
Chapter 3	If...Then...Else Statement	27
3.1	Relational operators	28
3.2	If...Then...Else statement	30
3.3	ElseIf keyword	32
3.4	Official syntax	34
3.5	Exercise	35
Chapter 4	Boolean Operations	36
4.1	Logical operators	37
4.2	Nested If...Then...Else statement	41
4.3	Exercise	42
Glossary	44

Chapter 1 Introduction

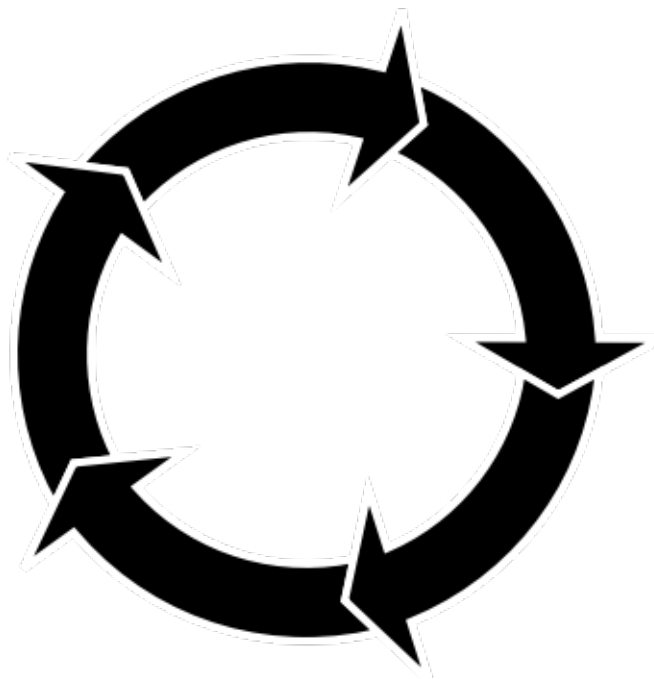
This book is an introductory course of **Visual Basic.NET**. Before we dive into the VB.NET, a few basic ideas of programming are introduced here.

What is programming?

Programming is the process of designing, writing, testing and debugging (remove mistakes from) a program. To write a program, you need to know and solve the problem first. After solving the problem, you tell the computer how to solve the problem using a **programming language**.

For example, if you need to write a program that calculates percentage changes, you should first solve the problem by obtaining the formula of percentage change. After that, you will write a program to do the task.

To get the idea of how to create a computer program, you may refer to the diagram below. (Note: after having a finished program, you can do an evaluation to look for possible improvements.)



What is a programming language

In computers, instructions are stored in **machine code**, that each instruction (e.g. ADD, SUBTRACT, MULTIPLY, etc.) is given a unique number. Computers can understand machine code only.

However, machine codes are very difficult to be understood by human. Therefore, **programming languages**, that are understandable by human, are created to bridge the gap.

A programming language consists of a **specification** and an **implementation**. The specification provides a definition of the programming language, and programmers write programs according to the specification. And the implementation is a translator: something that converts the program from the programming language into machine code.

Compilers and interpreters

Compilers are a type of translator that converts source code into an object program. The object program is stored in an executable file (.exe in Windows), which can be executed directly in a computer. The usage of a compiler can be summarized in the diagram below:



On the contrary, an **interpreter** reads the instructions one by one, and execute each instruction immediately after reading it. No object program is made by the interpreter. This approach may sound simpler, but its efficiency is generally much lower than compilers because the same program statement needs to be translated every time it is run.

Bytecode and virtual machines

In some programming languages, including Java and Visual Basic.NET, programs are not directly compiled to machine code. Instead, they are compiled to some intermediate representation called **bytecode**.

The generated bytecode is opened and executed by a **virtual machine (VM)**. A virtual machine is actually a translator of the bytecode, which consists of an interpreter or a just-in-time (JIT) compiler, or both.

Just-in-time compilation

Just-in-time (JIT) compilation means doing the compilation at run-time, instead of prior to execution. This is generally more efficient than interpreters because compilation is only done once. However, there is a small delay at the start of execution because of the compilation.

Java, JavaScript and .NET framework are a few popular systems that utilise JIT compilation.

1.1 Programming languages

Here are the names and characteristics of some programming languages.

NOTE

You are only required to remember the names of these programming languages. (The characteristics and example are provided for reference only.)

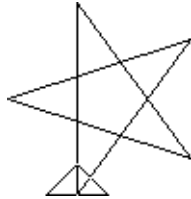
BASIC

BASIC (Beginner's All-purpose Symbolic Instruction Code) is designed to be easy-to-learn. It gradually evolved into modern programming languages, firstly QuickBasic, and then Visual Basic.

Example program	Output
<pre>10 FOR I=1 TO 5 20 PRINT I*I 30 NEXT I</pre>	<p>1 4 9 16 25</p>

Logo

Logo is known for drawing attractive graphics. It was used to teach programming in schools.

Example program	Output
<pre>REPEAT 5 [FD 100 RT 144]</pre> <p>(Note: FD = forward, RT = right turn. Only the star is drawn. The triangle is the cursor, pointing upwards.)</p>	

Pascal

Pascal was designed mainly to teach good programming style. Pascal is being taught in the HKDSE ICT syllabus. Some modern programs are also written in Pascal.

Example program	Output
<pre> program example1; var i : integer; begin for i := 1 to 5 do begin WriteLn(i*i); end; end. </pre>	<pre> 1 4 9 16 25 </pre>

C++

C++ is one of the most popular programming language (much more popular than Pascal). If you want to be a good programmer, C++ is a programming language that you must learn.

C language, the predecessor of C++, is also widely used today.

Example program	Output
<pre> #include <iostream> int main(int argc, char* argv[]) { for (int i=1; i<=5; i++) { cout << i*i << endl; } } </pre>	<pre> 1 4 9 16 25 </pre>

Java

Java is one of the most popular programming languages, mostly used in client-server or web applications. Java is also the programming language for Android apps.

Example program	Output
<pre>public class HelloWorld { public static void main(String []args) { for (int i=1; i<=5; i++) { System.out.println(i*i); } } }</pre>	1 4 9 16 25

JavaScript

JavaScript is available in a web browser, allowing dynamic user interaction in a web site. New web technologies, i.e. Ajax and HTML5 rely on JavaScript as the programming language.

There is also a framework called node.js that allows JavaScript code to be run outside browsers.

PHP

PHP is one of the most popular programming languages for a web server. With PHP you can dynamically generate web contents, and many other things, depending on user input. PHP is used in <http://www.tanghin.edu.hk>.

Visual Basic.NET

Visual Basic.NET is the programming language you are going to learn in this book. Visual Basic is the easiest programming language to learn if you want to create a program in Graphical User Interface (GUI).

Chapter 2 Console Applications

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

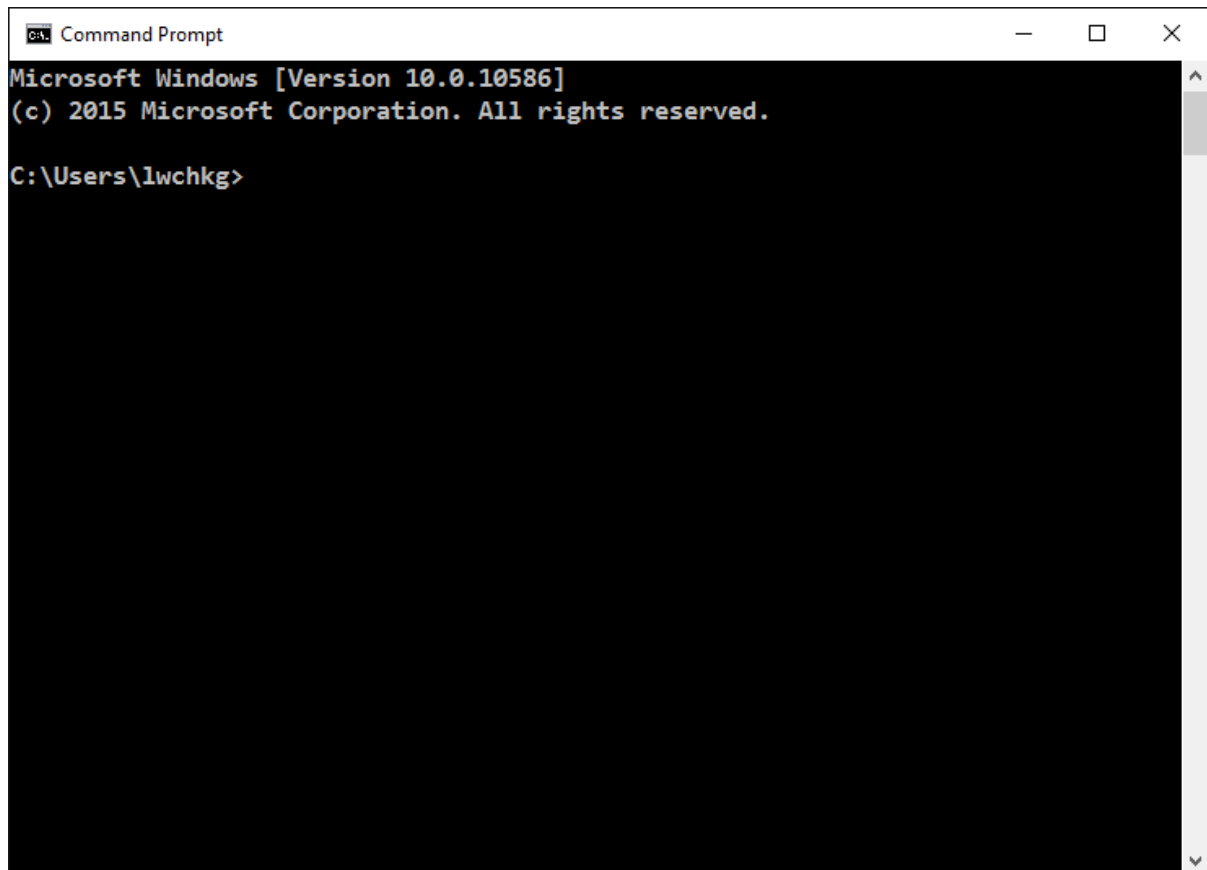


Figure 1. Command Prompt

Comparison with GUI applications

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

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

2.1 Your first VB.NET program

Now you will create your first VB.NET program.

Creating a new project

This book assumes that you are using Visual Studio 2015 Community. The user interface may be a bit different in other versions of Visual Studio. However, these small differences should not make other versions unusable.

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

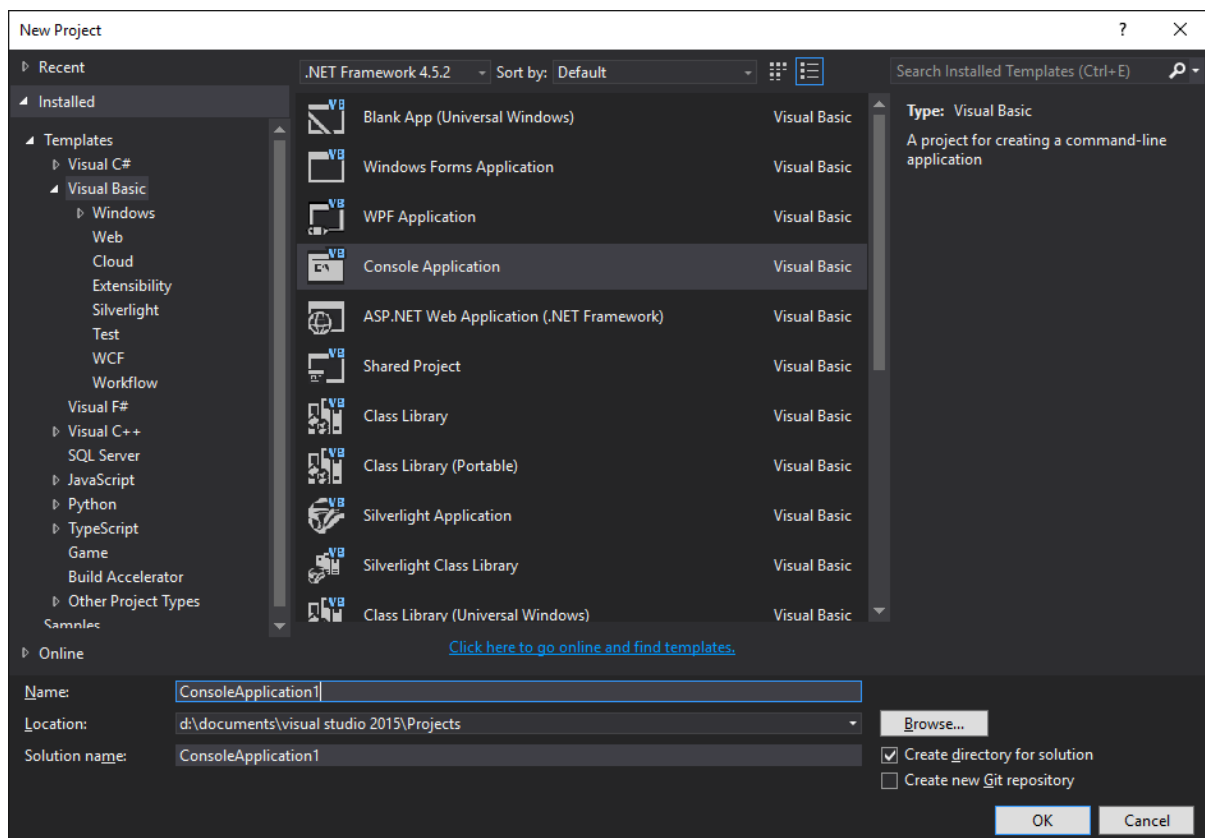


Figure 1. Creating a New Project

Unable to install Visual Studio?

You can make simple programs in Visual Basic.NET online in [TutorialsPoint](#) [CodingGround](#).

NOTE

When you are using online platforms, there is generally no need to add `Console.ReadLine()` at the end of the program.

Adding some code

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

```
Module Module1

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

End Module
```

Running the program

Press the key `F5` to run the program. The output should look like this:

```
Hello world!
```

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

Using the code view

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

- Remember the shortcut key `Ctrl + Z` for undo. This is the *FIRST* thing to do when something is wrong.
- Use the keys `Insert`, `Delete`, `Backspace` and `Enter`. Do this
 - at the beginning of a line, and
 - in the middle of a line.
- Select text by holding `Shift` while using the arrow keys.
- Use the shortcut keys `Ctrl + C` (copy), `Ctrl + X` (cut) and `Ctrl + V` (paste).
- Type when there is an active selection. See how the text is overwritten.
- Advanced: While holding `Ctrl`, press the left and right arrow keys to move the cursor. You may use this in combination with the `Shift` key.

2.2 The Console object

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

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

`Console.WriteLine` and `Console.Write`

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

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

The next example shows the differences of `Console.WriteLine` and `Console.Write`.

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

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

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

See the output below to check your understanding of the example.

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

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

Class work

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

Output	Visual Basic Program
This is my first console program. It is easy!	<pre>Sub Main() End Sub</pre>

Substitutions in the `Console.WriteLine` and `Console.Write`

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

Since the output string is the first argument, `{0}` is replaced by the content of the second argument, `{1}` by the third argument, etc. Here is an example:

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

And here is the output:

```
David and Peter are good friends.
9 * 9 = 81
```

Console.Clear

`Console.Clear` is used to clear the content of the console. After the console is cleared, the cursor goes to the top-left corner. `Console.Clear` is introduced in this book only for completeness, i.e. it is not used in this book.

Console.ReadLine

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

A typical use of `Console.ReadLine` is as follows:

```
Console.Write("Enter the number n: ")
Dim n As Integer = Console.ReadLine()
Console.WriteLine("The value of n is {0}.", n)

Console.ReadLine()
```

If the user enters “100” in the input, the output is:

```
Enter the number n: 100
The value of n is 100.
```

In this book, `Console.ReadLine()` is also used to prevent the program from closing.

TIP

If `Console.ReadLine()` is missing before the end of the program, the program is closed immediately after running. If you are using Visual Studio, you will not be able to read the output of the program.

2.3 Basic calculations in Visual Basic

Now we are learning how to store data to variables, and how to perform simple calculations.

Introduction to operators

To do calculations we use operators and brackets. Here is an incomplete list of operators:

Operator	Meaning	Example	Result
+	Addition	$3 + 8$	11
-	Subtraction	$10 - 15$	-5
*	Multiplication	$3 * 8$	24
/	Division (floating point)	$14 / 4$	3.5
^	Exponentiation	$3 ^ 4$	81

The usual order of operations applies here. **^** is calculated first, then ***** and **/**, finally **+** and **-**.

TIP

The * symbol, called asterisk, can be five-pointed, six-pointed or eight-pointed depending on the font. For handwriting, all of them are acceptable.

Class work

Evaluate the following expressions:

VB expression	Result	VB expression	Result
$3 + 6$		$3 - 6$	
$3 * 6$		$15 / 7$	
$2 ^ 5$		$5 ^ 2$	
$3 ^ (5 - 2)$		$3 + 4 * 5$	

Parentheses

Parentheses (or round brackets) are used to change the order of operations. Other kinds of brackets are not accepted in Visual Basic.

IMPORTANT

Do not forget to add a pair of parentheses for division.
E.g. $\frac{a}{b+c}$ should be $a/(b+c)$ but not $a/b+c$.

Class work

Convert the following formulae into Visual Basic code:

Mathematical formula	VB statement
$a = b + c$	<code>a = b + c</code>
$D = b^4$	
$x = 90(n - 1)$	
$p = \frac{a}{(1+r)}$	
$A = P \left(1 + \frac{r}{n}\right)^{nt}$	

2.4 Variables in Visual Basic

In Visual Basic, you use variables to store values. Variables have a name and a data type.

Declaring variables with Dim statement

Before you use a variable, you should tell Visual Basic in advance, using the `Dim` statement:

```
Dim variablename [As type]
```

The data type of the variable can be one of the following:

Data type	Description
Integer	Integer (between -2,147,483,648 through 2,147,483,647)
Single	Single-precision floating point (about 7 significant figures).
Double	Double-precision floating point (about 16 significant figures).
Boolean	Either True or False
String	Text (holds your name, ID card number, etc...)

See the following example for their use:

```
Dim statements  
Dim score As Integer  
Dim weight As Double  
Dim finished As Boolean  
Dim name As String
```

Class work

Determine the correct data type(s) for the following data, and write down a Dim statement, where the name of the variable is the underlined word.

Description of data	Data type	Dim statement
The <u>height</u> of a student (in cm).		
The <u>marks</u> of a test.		
The academic <u>grade</u> of a student.		
The <u>age</u> of your father.		
Whether you are <u>wealthy</u> .		

Keywords and names of variables

Some words have special meanings in Visual Basic, such as `Dim`, `As`, `Integer`, `Double`, `Boolean`, and `String`. These words are known as keywords, or reserved keywords. The list of Visual Basic keywords in Visual Studio 2015 can be found in [https://msdn.microsoft.com/en-us/library/dd409611\(v=vs.140\).aspx](https://msdn.microsoft.com/en-us/library/dd409611(v=vs.140).aspx) The name of a variable cannot be any of the reserved keywords.

In addition, it must obey the following rules:

- It must begin with an alphabetic character or an underscore (`_`).
- It must only contain alphabetic characters, decimal digits, and underscores.
- It must contain at least one alphabetic character or decimal digit.

Giving values to variables in the Dim statement

We can assign a value to a variable in the same Dim statement. This is a good practice in programming because it eliminates potential mistakes.

```
Dim score As Integer = 0
Dim weight As Double = 129.3
Dim finished As Boolean = False
Dim name As String = "Chan Tai Man"
```

Declaring several variables in a single line

We can declare several variables in a single Dim statement. However, in Visual Basic, we cannot assign a value to these variables at the same time.

```
Dim a, b, c As Single, x, y As Double, i As Integer
' a, b, and c are all Single; x and y are both Double
```

Using variables to store values

We can assign a value to a variable with the equal sign (=). Once a new value is stored in the variable, the old value is forgotten. See the example below:

```
Dim x, y As Integer
x = 5      ' x is set to 5
y = 3      ' y is set to 3
x = 2      ' x becomes 2, y is unchanged
y = x + 2  ' y becomes (x+2)=(2+2)=4, x is unchanged
x = y + 3  ' x becomes (y+3)=(4+3)=7, y is unchanged
```

Class work

With reference to the programs below, write down the values of the variables **after** the execution of each of the statements.

Program	Value of a	Value of b	Value of t
<code>Dim t As Integer</code>	---	---	---
<code>Dim a As Integer = 2</code>	2	---	---
<code>Dim b As Integer = 6</code>	2	6	---
<code>t = a</code>			
<code>a = b</code>			
<code>b = t</code>			

Program	Value of p	Value of q
<code>Dim p As Double = 3</code>		---
<code>Dim q As Double = 4.5</code>		
<code>q = p - 0.5</code>		
<code>p = 2</code>		
<code>p = (p + q) / 2</code>		
<code>q = (p - q) / 2</code>		

2.5 Writing console applications

You have learnt about the `Console` object, variables, and doing calculations in Visual Basic. Now you can write simple programs by using them together. See the example below:

```
Module Module1

    Sub Main()
        Console.Write("Enter the base of the triangle: ")
        Dim base As Double = Console.ReadLine()

        Console.Write("Enter the height of the triangle: ")
        Dim height As Double = Console.ReadLine()

        Console.WriteLine("The area of the triangle is {0}.",
                           base * height / 2)

        Console.ReadLine()
    End Sub

End Module
```

Here is how the program runs:

```
Enter the base of the triangle: 5
Enter the height of the triangle: 7
The area of the triangle is 17.5
```

There are a few main points:

- To show a message (prompt) before input, use `Console.Write` before `Console.ReadLine`.
- Output with one or more `Console.WriteLine` statements.
- If a statement is too long, you may break it into several lines to make it easier to read.

What if I do not enter a number?

If you do not enter a number in the input, e.g. `7a`, the program will end up in a runtime error like this:

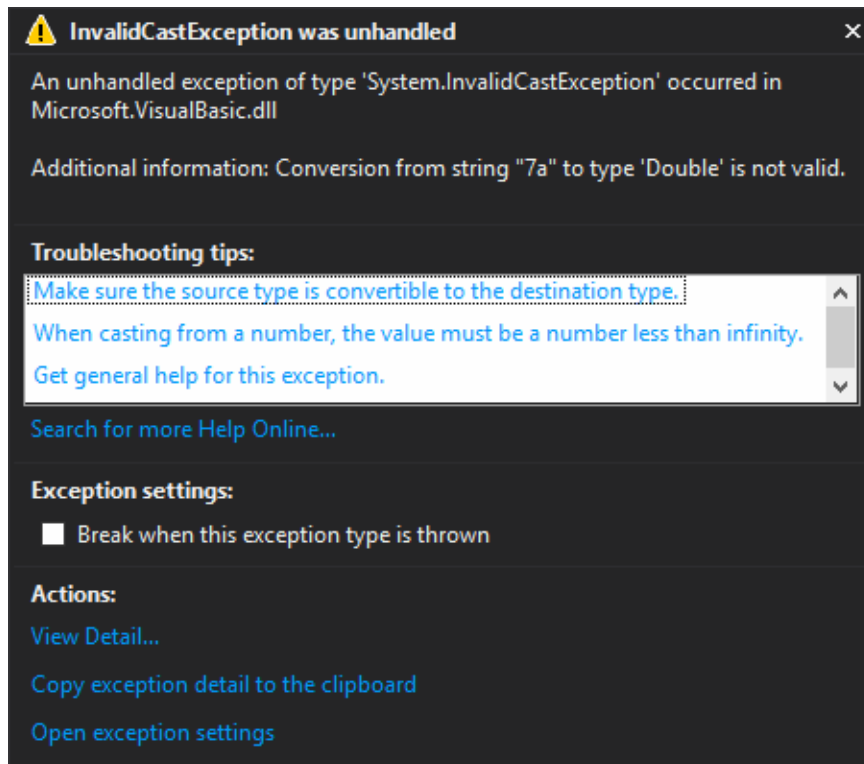


Figure 1. Runtime error: Conversion from string "7a" to type 'Double' is not valid.

The error happens because Visual Basic is unable to convert “`7a`” into a number. A good programmer should deal with these cases with additional code.

We are now not good enough to write these code properly. But we can suppress the error using the `Val` function, which does always succeed. All we need to do is like the example below:

```
Dim base As Double = Val(Console.ReadLine())
```

NOTE

Using `Val` to suppress warning has many undesired effects. You should used other ways to do the conversion after you learn them.

2.6 Coding style

Professional programmers work with projects with hundreds of source files. It is very important to keep programs maintainable, in particular easy to read.

Indentation

One of the standard practice is to indent the code inside each statement block, i.e. shift the code to the right. Fortunately, you do not need to know the details now, because Visual Studio indents your code automatically.

Comments

Comments are messages in the source code that are intended to be read by humans. They are ignored by compilers. In Visual Basic, comments start with a single quote ('), which is followed by any text you want. Here is an example:

```
Module Module1
    Sub Main()
        ' This is a comment that occupy for a whole line.
        Console.WriteLine("Hello world!") ' Another comment
        Console.ReadLine()
    End Sub
End Module
```

Appendix: Option Strict On

To produce good code in Visual Basic, it is advised to write `Option Strict On` on the first line of every `.vb` file. This forbids a few types of implicit conversions, including the implicit conversion from string to numbers. Once you can handle the conversion properly (using `Val` is not handling properly), you should use that options in your Visual Basic code.

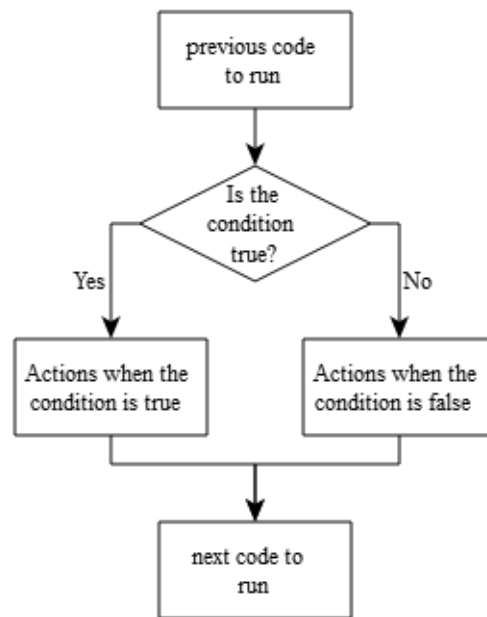
2.7 Exercise

1. Write a program to find the value of the n -th triangle number, i.e. $\frac{n(n+1)}{2}$.
2. Write a program to calculate the value of f in $f = \frac{uv}{u+v}$.
3. Write programs that calculates percentage changes. There should be one program for each:
 - the original value,
 - the new value, and
 - the percentage change. (Note: What are the formulas?)

Chapter 3 If...Then...Else Statement

We want computer programs to do more than just finding results of formulae. In this chapter, you will learn how to do **branching** (or make **decisions**) with If...Then...Else statements.

Branching means to perform different actions depending on a condition. The concept of branching is described in the picture below.



3.1 Relational operators

Relational operators (or comparison operators) are the equality signs and inequality signs in Mathematics. This is evaluated to one of the Boolean values. If the equality or the inequality is satisfied, then the result is `True`, otherwise the result is `False`. Here is the list of the relational operators:

Operator	Meaning	Example	Result
=	Equal to	9 = 11	
>	Greater than	11 > 9	
<	Less than	11 < 9	
>=	Greater than or equal to	15 >= 15	
<=	Less than or equal to	9 <= 15	
<>	Not equal to	9 <> 9	

Class work

Convert the following comparison into Visual Basic code:

Comparison	VB code
Is Peter (PHeight) taller than Mary (MHeight)?	
Is Linda's age (LAge) not equal to May's age (MAge)?	
The passing mark is 50. Has Gigi (GMark) failed the test?	
It is free to travel by MTR for a person with height 100 cm or below. Is it free for Kitty (KHeight) to travel by MTR?	

Law of trichotomy

For numbers a and b , exactly one of the following holds:

- $a > b$;
- $a = b$; or
- $a < b$.

This is known as the law of trichotomy. It holds for every kind of quantity that has order, including `Integer`, `String` and `Boolean` in Visual Basic.

However, there is a special value called NaN (not a number) for `Single` and `Double`. Since NaN is not a number, $a > b$, $a = b$ and $a < b$ are all false. Law of trichotomy works only if you ignore the case of NaN. In fact, comparison of floating point numbers is a complex topic of its own.

Logical opposites of relational operators

Every relational operator corresponds to one or two possibilities of the law of trichotomy. By picking up the remaining possibilities, we obtain the logical opposite of the relational operator.

Operator	Logical opposite
=	
>	
<	
>=	
<=	
<>	

NOTE

Logical opposites works only if the quantity under comparison has order.

3.2 If...Then...Else statement

An `If ... Then ... Else` statement conditionally executes a group of statements, depending on the value of an expression. (See the flowchart below.)

The basic syntax of `If ... Then ... Else` statement is listed here.

```
If condition Then
    Statement1
    Statement2
    ...
Else
    Statement3
    Statement4
    ...
End If
```

TIP

A **condition** is an expression that evaluates to a Boolean value, and this is often a single Boolean variable (with its value calculated before).

Sometimes we do not want to execute anything when the condition is false. In this case, we can skip the section starting with `Else`, i.e.

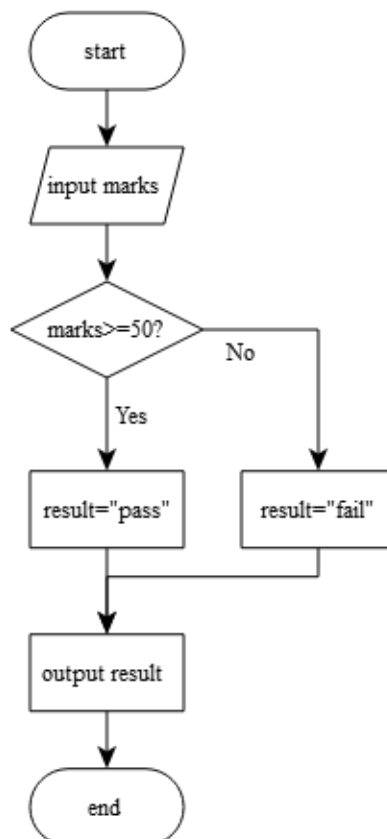
```
If condition Then
    Statement1
    Statement2
    ...
End If
```

The following example shows the use of `If...Then...Else` statements:

```
Console.Write("Enter the marks:")
Dim marks As Integer = Console.ReadLine()
Dim result As String
If marks >= 50 Then
    result = "pass"
Else
    result = "fail"
End If
Console.WriteLine("You have got a {0}.", result)
Console.ReadLine()
```

In the above example, if the marks is 50 or above, the output is “You have got a pass.” Otherwise (i.e. marks less than 50), the result is “You have got a fail.”

Here is a flowchart of the program:



3.3 Elseif keyword

To get more than two outcomes, additional conditions using the keyword `ElseIf` (else and if together in one word) can be added.

```
If condition1 Then
    Statement1
    ...
ElseIf condition2 Then
    Statement2
    ...
(more ElseIf conditionals if applicable)
Else
    Statement3
    ...
End If
```

To see how the keyword `ElseIf` works, see the example below:

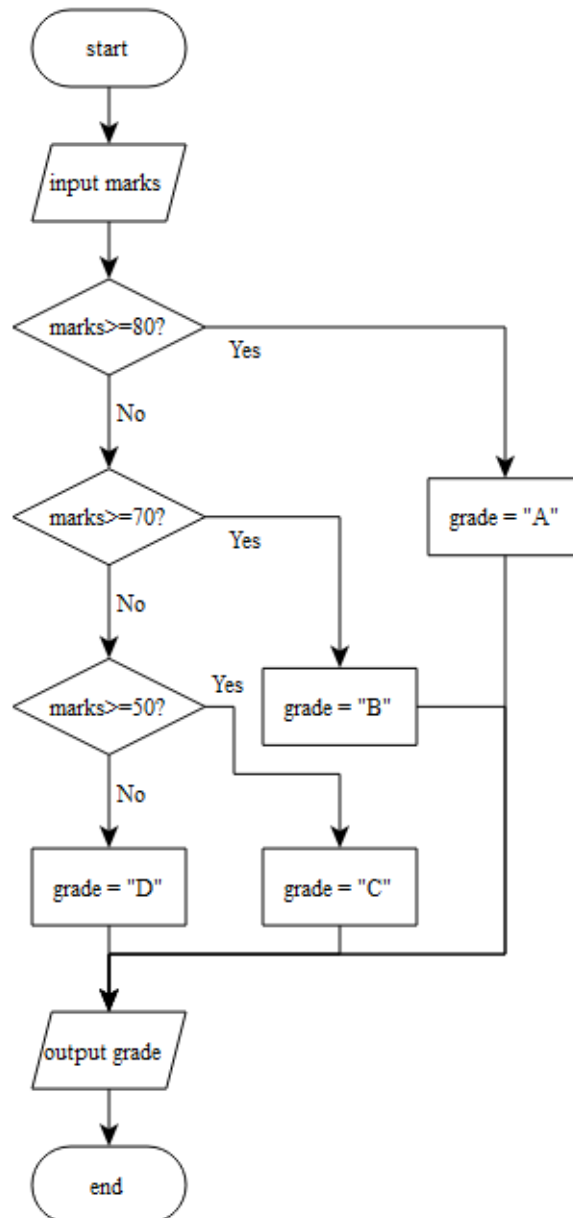
```
Console.WriteLine("Enter the marks: ")
Dim marks As Integer = Console.ReadLine()

Dim grade As String
If marks >= 80 Then
    grade = "A"
ElseIf marks >= 70 Then
    grade = "B"
ElseIf marks >= 50 Then
    grade = "C"
Else
    grade = "D"
End If
Console.WriteLine("Your grade is {0}.", grade)
Console.ReadLine()
```


In the example, the marks are used to determine the grade. When we check for the second condition, i.e. `marks >= 70`, we already know that `marks >= 80` is false. Therefore, only people with marks from 70 to 79 get a grade of “B”.

(Note: In this example, the marks are integers. This is NOT true in Tang Hin, where the marks are correct to 2 decimal places.)

Here is a flowchart of the program:



3.4 Official syntax

Here is the official syntax of the If...Then...Else statement. The parts inside [] are optional.

```
If condition [ Then ]  
    [ statements ]  
[ ElseIf elseifcondition [ Then ]  
    [ elseifstatements ] ]  
[ Else  
    [ elsestatements ] ]  
End If  
-or-  
If condition Then [ statements ] [ Else [ elsestatements ] ]
```

When you enter your code into the Visual Basic IDE, it will do the following changes automatically:

- Add the missing keyword `Then`
- Change `Else If` into `ElseIf`
- Change `EndIf` into `End If`

WARNING

If you are not using Visual Studio, do not write `Else If` or `EndIf`. They will result in compile errors.

And here is an example of the single line syntax. The symbol “:” is used to separate multiple statements. (Note: you may use “:” outside the If...Then...Else statement.)

```
If A > 10 Then A = A + 1 : B = B + A : C = C + B
```

3.5 Exercise

- Write a program in which the user enters a number. If $x^3 - x \geq 100$, output “A big number!”, otherwise output “A small number!”.
- Write a program in which the user enters the amount of pocket money he/she spends per week. The program will output a message according to the table below:

Pocket Money (\$)	Message
0	You didn't spend any money. Are you lying?
> 0 and < 100	Below average. What a good student!
≥ 100 and < 200	Average.
≥ 200	Above average. Consider to spend less!

- BMI Calculator:** The Body Mass Index (BMI) is a method to see if you are overweight or underweight. Your program should receive two inputs: mass (in kg) and height (in cm). Calculate the BMI by to the formula
$$\text{BMI} = \frac{\text{mass (in kg)}}{[\text{height (in m)}]^2}.$$

(Be careful with the unit!!!)

Finally, your program should determine the result according to the following table:

BMI	Classification
< 18.5	Underweight
18.5 – 23.9	Average
24.0 – 27.9	Overweight
≥ 28	Obese

(Note: what should be classification if BMI = 23.91?)

- Write a program that the user will enter his/her marks in Chinese, English and Mathematics subjects. Then output the number of subjects that he/she has failed (i.e. marks less than 50).

Chapter 4 Boolean Operations

In the last chapter, we have learnt to use If...Then...Else statement to make simple decisions. However, we do often make decision based on a combination of conditions.

For example, when we check whether an examination mark is valid, you will check whether the mark is between 0 and 100 (both inclusive). To do this we need two conditions:

```
marks >= 0 and marks <= 100.
```

And the **logical operator** `And` is needed to join the conditions together, i.e.

```
If marks >= 0 And marks <= 100 Then
```

4.1 Logical operators

In the last section, the word `And` is a logical operator in Visual Basic. Here three logical operators are studied: `Not`, `And`, `Or`. Here are their meanings:

Operator	Meaning	Syntax
Not	Negation. True/False are inverted.	<code>Not condition1</code>
And	See if all conditions are true.	<code>condition1 And condition2</code>
Or	See if one or more conditions are true.	<code>condition1 Or condition2</code>

Truth tables

To understand a logical operator, we construct a table to list its possible inputs and outputs. This table is known as truth table. Now complete the following truth tables by writing down the results (`True` or `False`).

Table 1. And operator

<i>a</i>	<i>b</i>	<i>a And b</i>
True	True	
True	False	
False	True	
False	False	

Table 2. Or operator

<i>a</i>	<i>b</i>	<i>a Or b</i>
True	True	
True	False	
False	True	
False	False	

Table 3. Not operator

<i>a</i>	Not a
True	
False	

Order of operations

The evaluation of expressions in Visual Basic is ordered by its operator precedence below.

11 (Highest)	Exponentiation	^
10	Unary identity and negation	(unary)+ (unary)-
9	Multiplication and floating-point division	* /
8	Integer division	\
7	Modulus arithmetic	Mod
6	Addition and subtraction	+ -
5	String concatenation	&
4	Relational/comparison operators	= <> < <= > >=
3	Negation	Not
2	Conjunction	And
1 (Lowest)	Inclusive disjunction	Or

TIP

Not is evaluated before And, and And is evaluated before Or. Add a pair of parenthesis if this is not what you want.

NOTE

Operators introduced in later chapters are also listed here. An unary operator has only one expression next to it, e.g. the minus in $(-3)^2$.

Class work

Convert the following to Visual Basic conditions. The names of the variables are underlined.

Comparisons	Visual Basic condition
If the <u>mark</u> is outside the range $0 \leq y \leq 100$.	
If the <u>guess</u> is between 5 and 9 (both inclusive).	
If the <u>day</u> is "Mon" or "Tue"	
If the <u>sex</u> is "M" and the <u>weight</u> is greater than 80 (kg).	
If the <u>age</u> is not less than 65, and <u>today</u> is "Sat".	

TIP

Inclusive means include the end values (5 and 9 in the question). The opposite of inclusive is exclusive.

Evaluate the following Visual Basic Expressions.

Hint: The result is either `True` or `False`.

VB expression	Result
<code>4 > 2 And 1 = 3</code>	
<code>20 < -5 Or 3 >= 3</code>	
<code>-5 <= -7 Or 3 / 2 > 2</code>	
<code>1 > -2 And 5 / 7 >= 3 / 5</code>	
<code>Not (5 ^ 2 >= 2 ^ 5)</code>	
<code>Not (6 <= -2 And 7 > 12)</code>	
<code>1 > 2 And 3 = 4 Or 5 <= 6</code>	
<code>Not True And Not False</code>	

Convert the following mathematical expressions to Visual Basic conditions:

Mathematical expression	Visual Basic condition
$0 < a < 18$	
$x \leq y \leq z$	
$a > 0 > b$	

Example

```
Console.WriteLine("Enter your age: ")
Dim age As Integer = Console.ReadLine

Console.WriteLine("Enter your sex (M/F): ")
Dim sex As String = Console.ReadLine

If age > 0 And age < 18 And sex = "F" Then
    Console.WriteLine("Welcome to the dreamland!")
Else
    Console.WriteLine("No entry.")
End If

Console.ReadLine()
```

Here is the output:

```
Enter your age: 17
Enter your sex (M/F): F
Welcome to the dreamland!
```


4.2 Nested If...Then...Else statement

You can put an If...Then...Else statement inside another statement block, such as another If...Then...Else statement. Please see the example below:

```
Console.WriteLine("Are you a member? (Y/N) ")
Dim s As String = Console.ReadLine()
If s = "Y" Or s = "y" Then
    Console.WriteLine("Members - Free of charge.")
Else
    Console.WriteLine("You are not a member.")
    Console.WriteLine("Input discount code (ENTER for none): ")
    Dim s1 As String = Console.ReadLine()
    If s1 = "5318" Then
        Console.WriteLine("Discount #1: the entry fee is $2.")
    ElseIf s1 = "2407" Then
        Console.WriteLine("Discount #2: the entry fee is $5.")
    Else
        Console.WriteLine("No discount: the entry fee is $10.")
    End If
End If
Console.ReadLine()
```

Here is the output:

```
Are you a member? (Y/N) N
You are not a member.
Input discount code (ENTER for none): 2407
Discount #2: the entry fee is $5.
```

TIP

Using nested If...Then...Else statements is bad for readability. To make programs readable, keep the use of nested statements to a minimum.

4.3 Exercise

1. Convert the following mathematical expressions to Visual Basic conditions:

Mathematical expression	Visual Basic condition
$50 \leq y < 60$	
$-1 < b < 1$	
$p \geq q \geq r > 0$	

2. Write down the following conditions without using the Not operator:

Condition	Without Not operator
Not (marks ≥ 0 And marks ≤ 100)	
Not (num < 0 Or num ≥ 10)	

3. Ask the user to input the age and sex. Determine the result based on the table below:

Age / Sex	M	F
≥ 18	Man	Woman
< 18	Boy	Girl

The output of the program should be the same as the following:

(Note: replace “man” by the words above.)

```

Enter your age: 18
Enter your sex (M/F): M

You are a man.

```

4. The examination of the Computer subject in Acme School consists of two papers: theory and practical. The full marks are 100 in both papers. Write a program to calculate the overall grades, which is listed in the following table.

Condition	Grade
Less than 50 marks in any of the papers (regardless of all other conditions)	Fail
Having 80 or more marks in both papers	Distinction
Having an average mark of 70 or above	Merit
Otherwise	Pass

The output of the program should be the same as the following:

```
Enter the mark of theory exam: 80
Enter the mark of practical exam: 65

The grade is Merit.
```

Glossary

Boolean

A data type that has only two values: true or false. This is named after the English mathematician George Boole.

bytecode

An intermediate representation of programs, designed for efficient execution by a virtual machine. Bytecode earned its name because its opcodes (operation codes) are one byte in size.

compiler

A computer program that transforms source code to executable programs.

condition

An expression that evaluates to a Boolean value.

console applications

Programs that interact with users using a text-only interface.

floating point

A form of representation of real numbers, which is used in computers and scientific calculators. Floating point can be used to represent very large and very small numbers.

graphical user interface

Interface that contains graphical elements, such as labels, text boxes and buttons.

interpreter

A computer program that directly execute source code without compiling it to machine code.

just-in-time (JIT) compilation

Compilation done during execution of a program. This is used instead of interpreters to improve performance.

logical operator

Operators that do calculations on Boolean values.

machine code

A set of instructions executed directly by a computer's central processing unit (CPU). Every processor family has its own instruction set.

programming language

A notation, or a “language”, which is used for writing programs.

relational operators

Operators that do comparisons. These are the equal sign and five inequality signs.

source code

Instructions to computers written in a programming language.

virtual machine

A program that runs a certain kind of bytecode, or a program that emulates a certain kind of computer system.

Visual Basic.NET

Visual Basic was a programming language and IDE created by Microsoft, for which GUI programs can be written easily. Later it has been evolved into Visual Basic.NET, which contains feature of modern programming languages.