Zig Zag Education

Computer Science KS4/5

C# Programming Guide

Using Console Mode

Download support files from zzed.uk/productsupport

zigzageducation.co.uk

POD 9489

Contents

Teacher's Introduction

This resource was produced with the intention of helping teachers guide students to program in C# console mode. Alternatively, it can be used more independently by students to acquire skills in C#.

It covers **most*** of the areas that students would require at A Level, and **all areas** that would be required for GCSE and AS.

** Does not cover advanced data structures (stacks, queues, trees, graphs, dictionaries vectors or hash tables)*

There are ten chapters; with Chapters 1–9 focusing on procedural programming, and the final chapter serving as an introduction to object-oriented programming concepts.

Each chapter contains explanations to enable students to understand the concepts and syntax used. Interspersed within each chapter are clearly-labelled *Coding*, *Written* and *Consolidation* tasks, enabling students to demonstrate their understanding and practice the relevant skills.

Answers to tasks requiring written responses are provided in print on pages 83–84.

For the coding tasks, **completed exemplar code** is provided as a download via the ZigZag Education Product Support system, but please note that these are only offered as examples – other ways of solving the problems will exist. Generic marking guidance which could be relatively applied to all programming tasks is provided on page 85.

A C# *Quick Syntax Guide* is provided on pages 74–82, which can be used as a quick reference guide or used for classroom posters. *I would like to acknowledge and give thanks to Bert Billard for compiling and contributing this section.*

October 2019

Supporting .txt files containing answers to coding tasks are provided on the ZigZag Education Product Support system, which can be accessed via **zzed.uk/productsupport**

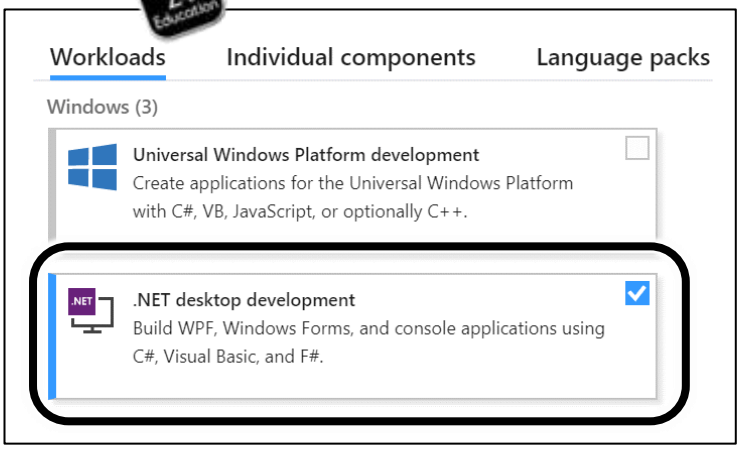
Chapter 1 – Welcome to

C# is an object-oriented language that enables developers to build a variety
the **.NET Framework**. You can use C# to create Windows applications, We
applications, client–server applications and database applications, to name
can use C# and Unity to create 2D and 3D games. In order to teach some c
programming through C# this guide will use the console mode application.
where output is to a plain command line type interface.

Downloading a C# IDE

An IDE is an integrated development environment – a software application
development. The IDE shown in this guide is Visual Studio Community 201
available at https://visualstudio.microsoft.com/vs/community/ After downloa
you install the .NET Desktop Development (see Image 1 below) as you will
correct project templates

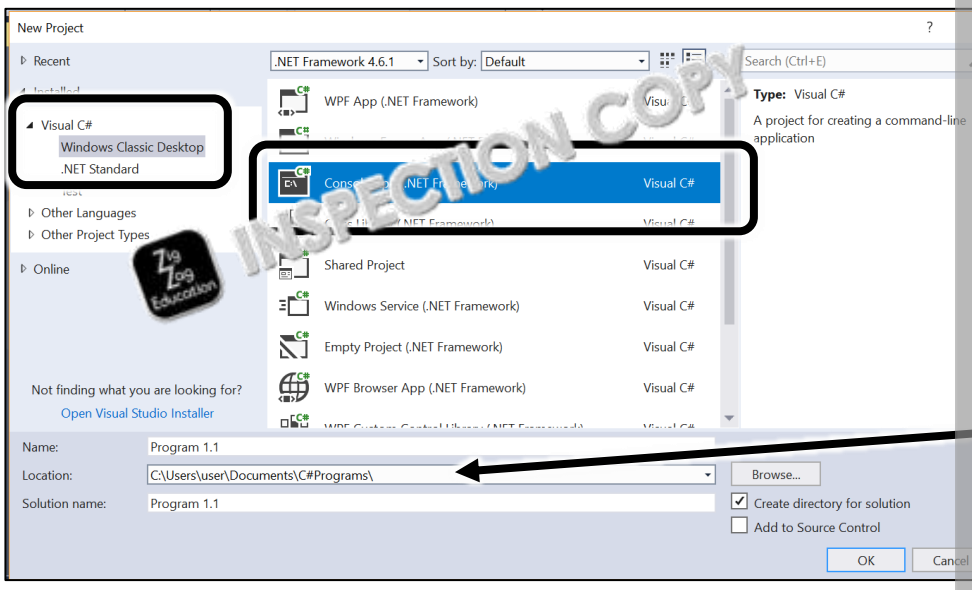
Image 1 – Visual Studio Installer



Opening a new console mode application

Before you can start to code, you need to open a new console mode applic
the Start Page or by going to the File menu and selecting New – Project. It
language and mode (C# Windows Classic Desktop) and then a project type
below shows how this is done.

Image 2 – How to choose a new console mode application



The console mode application template

Image 3 below shows the console mode application template.

Image 3 – console mode application template

Program 1.0 | Program_1._0.Program

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace Program_1._0
8   {
9       class Program
10      {
11          static void Main(string[] args)
12          {   Until you get to lesson 7 – the code we write will go
13          }   between these two curly brackets
14      }
15  }
```

usi
oth
na
na
org
co
cla
the
of
sta
pro
mo
the

Blocks of code (such as namespaces, classes, procedures, loops and s
defined using curly brackets {} – one to open the block and one to close t

Coding Task: *Program 1.1 – Outputting to the console*

Complete the program below and run it. Try changing the string ("hello Wor

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("hello World");
            Console.ReadLine();

        }
    }
}
```

```
hello World
```

A line of code that comp
a **statement**. Each state
with a semicolon.

Note the curved bracke
Console.WriteLine. V
to **pass data or argum**
this example, the string
is being passed to the W
String data is enclosed

We can start the execution of our program by clicking the Start button on th

The `Console.ReadLine()` statement at the end of the `Main` procedure is a
console window from closing immediately the program finishes executing s
also used to read data into our program from the console window as we se

Coding Task: *Program 1.2 – Input and output*

A more likely scenario for a program is that we will need to input and outpu

Pseudocode

OUTPUT "What is your name?"

INPUT name

OUTPUT name

This can be achieved in two different ways (see below) – try each method.

Program 1.2 v1

Try changi e question to achieve different outputs.

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("What is your name?");
        Console.WriteLine("Hello, your name is " + Consol
        Console.ReadLine();
    }
}
```

In C# we use the + symbol t strings together. In this case your name is" will be joined console window whatever th response to the question "W

Program 1.2 v2

This version of the same program uses a **variable** to store the input name. location in memory that is used to hold values used in our program that mig more about variables in Chapter 2.

```
class Program
{
    static void Main(string[] args)
    {
        string name;
        Console.WriteLine("What is your name?");
        name = Console.ReadLine();
        Console.WriteLine("Hello, your name is " + name);
        Console.ReadLine();
    }
}
```

Here we are using a string variable called 'name' to read in from the console window in response to the is later joined (concatenated) with "Hello, your nam console window.

Concatenating strings using placeholders

In the previous programs we used the + operator to join strings together. W
multiple strings, it may be easier to use the placeholder method.

Method 1 – joining strings using the + operator

```
Console.WriteLine("Hello, your name is " + name);
```

Method 2 – joining strings using a placeholder for variable values

```
Console.WriteLine("Hello, your name is {0}" , name);
Console.WriteLine("Hello, your name is {0} {1}" , firstnam
```

The numbers in the curly brackets refer to the contents of the variables stat
{0} is the first declared variable, **{1}** is the second declared variable, etc.

Coding Task: *Program 1.2*

Try changing program 1.2 so that it reads in a first name and a surname an
placeholder method to produce the output.

Pseudocode
OUTPUT "What is your first name?"
INPUT first name
OUTPUT "What is your surname?"
INPUT surname
OUTPUT "Your name is" and first name and surname

Adding comments to a program

Adding comments to a program allows us to explain, either to ourselves or
works, and is useful for explaining tricky bits of code.

```
// the statement below joins two strings together
Console.WriteLine("Hello, your name is {0} {1}" , firstnar
```

To add comments, we simply type two forward slashes followed by the c
The IDE software ignores these lines when compiling code for execution.

Coding Task: *Program 1.2 (continued)*

Add comments to program 1.2 to explain how your code works. Try to imag
to a non-coder how your statements are working in the program.

Reading in non-string values and casting data types

So far in the programs built we have only read in string (text) values. Obvio
have to deal with many types of data (integers, strings, Boolean, characters
this topic will be covered in more detail in Chapter 2, let's see what happen
when we need to read in from the console window non-string values.

```
static void Main(string[] args)
{
    int age;
    Console.WriteLine("enter your age");
    age = Console.ReadLine();
    Console.WriteLine(age);
    Console.ReadLine();
}
```

This progra
execute (a
indicates a
program is
string value
window) to

The string value th[illegible] needs to be **converted** to the data type of the va
it to. See t[illegible] example below.

```
age = int.Parse(Console.ReadLine());
```

Alternatively, the **`Convert`** method can be used for many different data type
converts to a 32-bit integer.

```
age = Convert.ToInt32(Console.ReadLine());
```

Coding Task: *Program 1.3 – Retro Sales*

Study the scenario below and the pseudocode provided, then build a progra
calculate the total price of a car.

Retro Sales is a local car dealership which sells a range of retro cars. It wa
***provide customers with the total price of a new car**. **VAT is charged** at*
cost of the car.

Models Sold:	Retro Original (£16,500) / Retro Panoramic (£17,50
Trim Package Costs:	Silver (£2,500) / Gold (£1,950)
Car Tax:	£30 for 12 months

Pseudocode

```
OUTPUT "What model do you want?"
INPUT Modeltype
OUTPUT "Please enter the pri[illegible] model"
INPUT Modelprice
OUTPUT "[illegible]t t[illegible]package do you want?"
INPUT Tr[illegible]kage
OUTPUT "What is the cost of your trim?"
INPUT Trimprice
CarTax = 30
Price = ModelPrice + Trimprice + CarTax
OUTPUT "The model you have chosen is " and model name
OUTPUT "The trim you have chosen is " and Trim name
OUTPUT "The price for your car is " and price
```

Program 1.4

Sarah makes curtains for a living. She has asked for a calculator that enabl
material she needs to make a set of curtains for a particular window, and h
material will cost her.

Study the code for the program below and answer the questions underneat

```
static void Main(string[] args)
{
    int windowheight;
    int windowwidth;
    int fullness;
    int materialneed
    int pr

    Conso  riteLine("What is the window height?");
    windowheight = int.Parse(Console.ReadLine());
    Console.WriteLine("What is the window width?");
    windowwidth = int.Parse(Console.ReadLine());
    Console.WriteLine("What fullness is required?");
    fullness = int.Parse(Console.ReadLine());
    materialneededmetres = windowheight * windowwidth * fullness
    Console.WriteLine("What is the price per metre?");
    price = int.Parse(Console.ReadLine());
    Console.WriteLine("Hello, you need {0} metres of material" ,
    Console.WriteLine("The price of your curtains is £{0}", mater
    Console.ReadLine();
}
```

The program **produces this output in the console window** when the curt
and the width is 2 metres. The fullness required is double the width, and th
per metre.

```
What is the window height?
4
What is the window width?
2
What fullness is required?
2
What is the price per metre?
2
Hello, you need 16  f material
The price  y  tains is £32
```

1. Explain how the program outputs to the console window in this progran

   .....

   .....

   .....

2. Explain how values typed in by the program user are input into the prog

   .....

   .....

   .....

3. Why have some in used the **.parse** method?

   .....

   .....

4. Sarah wants to make curtains for a window with the following values:

   Window height: 8 metres
   Window width: 3 metres
   Fullness: 2
   Material price: £4.67

   When the program runs with these values it crashes.

   Explain why this happens and suggest ways to avoid it. (TIP! You may build the program to see what happens.)

   .....

   .....

   .....

5. This question will involve you **building program 1.4 using the text fil** to calculate the VAT payable on the curtains – **add some code** to this job for her. The final output should now be the price plus the appropriat

   VAT should be calculated as 20% of the pric

Chapter 2 – Using Variab

What is a variable?

Variables are like containers. We use them in our programs to hold the valu
changing values during the course of a program and possibly to output ther
the end of, a program. In C#, when we declare a variable we need to state
be contained within it.

Here we have a **variable called X**.
X stores **integer** values and currently has
the value of **23** assigned to it.

Variable data types

The most common data types that you will use in C# programs are listed be
these can be found easily on a multitude of help websites using the search

Data type	Description	
String	A composite data type – strings are composed of ASCII characters	st
Integer	Integer data types store whole numbers in the range -2,147,483,648 to 2,147,483,647	in
Real	Real data types store numbers with a fractional part – there are a number of different data types for real numbers in C#	do fl de
Char	A char data type stores a single ASCII character	ch
Boolean	This data type will only store true or false	bo

Written Task

1. Find two more data types used in C# that are not listed in the table on t
   Describe and give an example of the C# syntax for each.

2. A teacher plans a program to store her students' mock exam marks. W
   teacher choose to store the values listed below?

   a. The student's surname

   b. The student's raw score

   c. The student's percentage mark

   d. The student's grade (A, B, etc.)

   e. Whether or not the student has passed the exam

Declaring variables

It is common practice to declare variables before using them. This serves th
place in memory for that variable and the values that will be assigned to it. I
by stating the data type followed by the identifier. The identifier is the name
will normally be reflective of the values stored in the variable. It is good prog
meaningful identifier names for variables and constants.

```
int examark;
```

Data type

Identifier

Initialising variables

Initialising variables is giving them their starting or initial value. This can be
declaration, or afterwards.

```
int examark = 32;
```

Here, the integer variable is bo
initialised in the same statemen

```
int examark;
examark = 32;
```

Here, the same integer variable is de
statement and initialised in a subsequ

Assigning values to variables

A key aspect of the definition of a variable is that it is a value that can be ch
for a variable to be assigned a new value (perhaps many times) during the

```
// a program to find and output the percentage score gained by a student
// paper is marked out of 60
int exammark = 0;
double exampercentage = 0;
Console.WriteLine("What is the exam mark gained");
exammark = int.Parse(Console.ReadLine());
exampercentage = exammark / 60 * 100;
Console.WriteLine("Your raw score was {0} and your percentage score was {1
Console.ReadLine();
```

In the program above, we can see that both exammark and exampercentag
default value of 0 (since it is not possible for a student to gain less than 0 in
assigned a different value after the input of the exam mark by the user and
percentage has been worked out.

Assignment operations

There are a number of different assignment operations that can be used in

Operation	Example	
=	`exampercentage = exammark / 60 * 100;`	This is s of the rig statemen hand par

+=	`int total = 4;`
	`total += 10;`
	
	the value of total after this operation will be 14
		Adds the assigns e.g. A+=

-=	`int total = 4;`
	`total -= 1;`
	
	the value of total after this operation will be 3
		Takes th assigns e.g. A-=

*=	`int total = 4;`
	`total *= 2;`
	
	the value of total after this operation will be 8
		Multiplies assigns e.g. A*=

/=	`int total = 8;`
	`total /= 2;`
	
	the value of total after this operation will be 4 (as 8 divided by 2 = 4)
		Divides t assigns e.g. A/=

%=	`int total = 8;`
	`total %= 3;`
	
	the value of total after this operation will be 2 (as this is the remainder when 8 is divided by 3)
		This (%) compute division value on e.g. A %

Arithmetic operators

Operator	Example	Expla
+	`Total = num1 + num2`	Adds two values together
*	`Total = num1 * num2`	Multiplies two values togeth
/	`Total = num1 / num2`	Divides two values
-	`Total = num1 - num2`	Subtracts one value from a
%	`Total = num1 % num2`	Finds the remainder from a
++	`num++`	Increments by 1 (so here, n
--	`num--`	Decrements by 1 (so here, away from its value)

Arithmetic operators and precedence

When calculations using variables or literal values, the rules of mathematica to as **BODMAS** apply. This means that operations are performed in the fol

1. Brackets (parts of a calculation inside brackets always come first)
2. Orders (numbers involving powers or square roots come next)
3. Division
4. Multiplication
5. Addition
6. Subtraction

Written Task

Build the program below and answer the questions that follow.

```
const int A = 10;
const int B = 4;
const int C = 6;
Console.WriteLine("B + C * A = {0}", B + C * A);
Console.WriteLine("B + C * A = {0}", (B + C) * A);
Console.ReadLine();
```

1. What is the difference in the output from the two WriteLine statements?
2. Why is there a difference?
3. Find out what the rules of precedence are that are applied to calculation

The VAR keyword in C#

Generally, it is considered good practice to declare a variable with its data t However, C# provides a facility that enables the compiler to determine automatically based on the value or expression assigned to it.

```
var amount = 23;
var surname = "Smith";
```

In the above examples, since the value assigned is an integer and string re will be declared as those types automatically. You cannot use the **var** keyw to it first, so the following statement would produce a compiler error:

```
var amount;
amount = 23;
```

Constants

These are used when the user does not provide the value and you don't wa
Therefore, the value of a constant does not change during execution and ca
value it is initialised with during execution of the program.

A constant must be declared and initialised at the same time.

```
const int A = 10;
const int B = 4;
const int C = 6;
```

Variable scope

Scope is the term used to show w e a variable can be used or seen. If a
memory location is used a ta variable. When the block of code in which
stops execu ti s m ory is released and that variable is no longer avai

Global var
These types of variable are accessible from within any function or procedur
there really is no such thing as a true global, although it is possible to use a
within a class that will act in a similar way. A static variable is one where the
entire run of the program.

Local variable
It is initialised and used only within the block of code in which it is declared.
procedure or function, or may be inside an IF block or loop block. When var
procedure or function, that local variable may be passed as a parameter to
called (see more in the chapter dealing with procedures and functions).

```
class Program
{
    public static double total;

    static void Main(string[] args)
    {
        double salary;
        Console.WriteLine("please enter your salary");
        salary = Convert.ToDouble(Console.ReadLine());
        workouttax(salary);
        Console.WriteLine("your salary after ax is {0}", total)
        Console.ReadLine();


    }

    stat ble workouttax(double salary)
    {
        const double tax = 0.20;

        total = salary - salary * tax;
        return total;
    }
```

total can be accessed and
procedure or function within cl

salary can only be
because it has been
when workouttax

tax can only be a
within the workou

Coding Tasks

Program 2.1: *Adding Two Numbers Together*

Write a simple program that will add two numbers together. The output shoul
below. It should work for any name and any input numbers, including those w

```
Hello and welcome to the number calculator - what is your
Mary
 please enter your first number Mary
34.56
 please enter your second number Mary
44.5
Thank you Mary your answer is 79.06
```

Program [illegible] *Cylinder Program*

Write a simple program that will find the volume and surface area of a cylin
constant. The image below shows the expected output to the console windo

```
Welcome to the cylinder program
please enter the radius of the cylinder
3
please enter the height of the cylinder
7
The volume of your cylinder is 197.9203338
The surface area of your cylinder is 188.495556
```

Program 2.3: *Working Out the Area of a Triangle*

Write a program to work out the area of a triangle when given the length of
formula. The function `Math.Sqrt()` can be used to find the square root of a

```
Please enter the length of the first side
5
Please enter the length of the second side
5
Please enter the length of the third side
5
The area of this traingle is  10.8253[illegible]055
```

Program 2.4: *Fence Panels*

Write a program that will work out how many fence panels are required for a

For the purposes of this program, garden fence panels come in a standard
program will first need to work out the perimeter of the garden that needs fe
calculating how many panels are required.

The image below shows the required output (although your program should w

```
Hello and welcome to the garden fence calculator
What is the width of your garden?
7
What is the length of your garden?
3
You will need 9 panels for your garden fence
```

Evidence required:

1. Annotated code listing for the above
2. Screenshots of the following tests:
   a. Where the garden length is 9 and the garden width is 3
   b. Where the garden length is 10.75 and the garden width is 5.9

Now answer the following questions.

1. Why might it be appropriate to use a constant for the fence panel width

   .....

   .....

   .....

2. How are constants different from variables?

   .....

   .....

   .....

   .....

3. Why is d programming practice to use meaningful identifier names
   constants?

   .....

   .....

   .....

   .....

4. Name three operations that might occur with variables during the cours

   1. .....

   2. .....

   3. .....

5. Why is it necessary to declare variables?

   .....

   .....

   .....

   .....

6. What does scope have on a variable?

   .....

   .....

   .....

   .....

7. A programmer has been asked to write a program to check the validity
   computer system. The user will have only three attempts before being
   system will check the entered password against the stored password.
   identifiers and data types for the following variables:

   i. A variable to store the entered password

      .....

      .....

   ii. A variable to store the saved password

      .....

      .....

   iii. A variable to store the number of unsuccessful attempts

      .....

      .....

   iv. A variable to store whether or not the entered password match
   password

      .....

      .....

Chapter 3 – Using Select

What is selection?

Selection refers to the fact that one or more lines of code may or may not b
outcome of a condition (test) being true or false. Selection statements use

```
if (score >= 40)
{
    Console.WriteLine("Congratulations !! You have passe
}
```

This conditional statement checks whether the value of score is more than
either true or false. If true, then the message in the curly brackets will be
happens in this program.

Why we need selection in programming

So far, the programs that we have seen have followed a series of steps in a
from beginning to end. Real-life programs obviously need more versatility a
need the ability to branch off in different directions, depending on values or
this ability to build decision-making into our programmed solutions to proble

A simple example might be the need in a password checker program to out
entered password is correct and a different message if it is incorrect.

Relational operators for use in selection

C# uses a number of different relational operators when constructing condit

Relational operator	Explanation	
==	Equal to	if (score ==
>	More than	if (score > 4
>=	More than or equal to	if (score >=
<	Less than	if (score < 4
<=	Less than or equal to	if (score <=
!=	Not equal to	if (score !=

Types of selection construct and C# syntax

C# has a number of different selection constructs.

IF…
This statement is used when you only want code to execute if the outcome
no code is to be executed when the outcome of the condition is false. In thi
execution will continue with any code statements outside the IF block.

The example below shows how an IF statement is written.

```
if (score == 40)
{
    Console.Write("you have achieved
}
```

IF keyword is followed by the condition
in curved brack

The cor on is followed by a set of curly brackets which contains the c
outcome of the condition is true. In this case, the message will be outpu
equal to 40.

Coding Task: *Program 3.1*

Write a program that will find the highest number out of an unsorted range
The screenshot below shows the expected output.

```
A program to find the highest number in an unsorted range
enter your first number
23
enter your second number
31
enter your third number
14
the highest value is 31
```

Coding Task: *Program 3.2*

Write a program that whether a given integer number is odd or e
The screen ows the expected output.

```
this is a program to find out if a given number is
please enter your number
13
The number 13 is an odd number
```

IF ELSE…
This type of selection construct has code to be executed if the outcome of t
alternative block of code if the outcome of the test is false.

```
if (score >=40)
{
    Console.WriteLine("you have passed the e
}
else
{
    Console.WriteLine("you have NOT passed t
}
```

Keyword **if** followed by the conditio
curved brackets.

The condition is first followed by a s
brackets which contains the code to
the outcome of the condition is true.

The **else** keyword is used to indicat
code following is to be executed if th
condition is false.

Logical operators for joining conditions

Logical operator	Explanation	C
&&	AND	(numA ==1 && numb
||	OR	(numA ==1 || numA =
!	NOT	(numA != 2)

Coding Task: *Program 3.3*

Write a program that checks whether a username and password used to lo
are both correct. The correct username is 'KGrid34' and the password is 'K

If the user attempt to log in is successful, then output a message 'you are lo
message 'The username or password is incorrect – log-in failed' should be

The screenshot below shows the expected output for an incorrect passwor

```
 welcome to the log-in system
 Please enter your username
KGrid56
 Please enter your pas
hhhhhhh
The use me password is incorrect - log-in fai
```

IF ELSEIF ELSE…
The ELSEIF part of the statement is used to add further conditions and alter

```
if (score >= 90)
{
    Console.WriteLine("you have achieved a distinction");
}
else if(score >=70 && score <90)
{
    Console.WriteLine("you have achieved a merit");
}
else if (score >= 40 && score < 70)
{
    Console.WriteLine("you have achieved a pass");
}
else
{
    Console.WriteLine("you have NOT achieved a pass");
}
```

The **else if** keywords are followed by a new co
block of code to be executed only if the outcom

The else statement at the end of an **if** and one o
provide code to be executed if none of the condi

Coding Task: *Program 3.4*

Write a program that takes in a student's marks for an exam and outputs th
mark. An A grade needs 80 or more marks, a B grade needs 70 or more ma
more marks, a D grade needs 45 or more marks, and an E grade needs 30

SWITCH CASE
SWITCH is used when you have either multiple cases with a particular outp
be executed depending on the case.

```
switch (monthnum)
{
    case 1:
        Console.WriteLine("January");
        break;
    case 2:
        Console.WriteLine("February");
        break;
    case 3:
        Console.WriteLine("March");
        break;
    defa
        Console.WriteLine("It must be one of the other months");
        break;
}
```

monthnum is the value to be passed into
– it is this value that is checked against

In this case, if **monthnum** =
output 'February' to th

Written Task

1. What is the purpose of the **break** keyword in the **switch** block of co
2. What is the purpose of the **default** keyword in the **switch** block of

Coding Task: *Program 3.5*

A department store offers various discounts, based on the following discou

- (ST – student – apply 10% discount)
- (SN – senior citizen – apply 12.5% discount)
- (CH – cardholder – apply 20% discount)
- (CN – discount voucher – apply 7.5% discount)

Your program should accept the net sale cost as an input and output the fi discount has been applied, based on the value of an input discount code. N if the customer is not entitled to a discount. You must use a switch stateme

The console window examples below show the expe output.

```
what is the cost of your goods
100
are you entitled to a concessio - enter true for yes and false for n
true
please enter your c type
Enter ST i re a student
Enter SN i re a senior citizen
Enter CH if are a cardholder
Enter CN if you have a discount voucher
SN
As a senior citizen the cost of your goods is 87.5
```

```
what is the cost of your goods
100
are you entitled to a concession discount - enter true for yes and false for
false
the undiscounted cost of your goods is 100
```

Multi-condition statements

In the program below, a random number is calculated and then checked to se If it is any number other than 6, 45 or 76, then a winning message is output they have not won anything. In this program, the condition is composed of t are joined together using the logical AND operator **&&**.

```
static void Main(string[] args)
{
    Random rnd = new Random();
    int luckynumber = rnd.Next(1, 100);
    Console.WriteLine("Welcome to the lu number progr
    Console.WriteLine("your lucky  be is {0}", luckyn

    if (luckynumber  uckynumber!= 45 && luckynum
    {
        iteLine("you are the lucky winner of a
    }

    else
    {
        Console.WriteLine("you have not won anything");
    }
    Console.ReadLine();
```

Program 3.6: *Insurance program*

Insurance Plus is an insurance broker based in Staffordshire. It offers disco
has had the program **"Program 3.6.txt"** written for it and now wants you to
Copy the existing program into a C# console mode project and run.

Now add the following amendments

1. The basic premium will be £330. This should be stored in the program.

2. The program should calculate the customer's premium based on their a
under 25 are charged double the basic premium. Customers who live i
to a 10% discount on their premium. Customers over 25 are charged th
who live outside the ST postcode area do not receive a discount.

Evidence required

1. Annotated code listing for the amended program.

2. Testing screenshots of the following tests:
    1. A customer under 25 with an ST postcode
    2. A customer over 25 with an ST postcode
    3. A customer under 25 with a BR postcode
    4. A customer over 25 with a PR postcode

Chapter 4 – Looping

The need for repetition in programming

Very often there is a need in programs to carry out repetitive tasks or actior

The simple example below of repeating lines of identical text shows that wit programming the amount of code we might have to write would be enormou flexibility of conditional repetition.

Loops of various kinds allow us to repeat sections of code without the need code repetitively.

```
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("this is a repeating program");
        Console.WriteLine("this is a repeating program");
        Console.WriteLine("this is a repeating program");
        Console.WriteLine("this is a repeating program");
        Console.WriteLine("this is a repeating program");
        Console.WriteLine("this is a repeating program");
        Console.WriteLine("this is a repeating program");
        Console.WriteLine("this is a repeating program");
        Console.WriteLine("this is a repeating program");
        Console.WriteLine("this is a repeating program");
        Console.ReadLine();
    }
}
```

Types of loop: the FOR loop

The FOR loop is known as a fixed or unconditional loop. It will repeat the co the specified number of times. It is composed of a number of features.

```
for (init; condition; increment)
{
    Code to be repeated goes here
}
```

init – the loop counter. It is a variable that holds the number of iterations (loops) that the following statements (in the brackets) will make.

condition – evaluates the value of the counter, and the body of the loop is executed if the condition is true.

A worked example

```
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            for (int i = 1; i <= 10; i++)
            {
                Console.WriteLine("this is a repeating program");
            }

            Console.ReadLine();
        }
    }
}
```

The progra[illegible] ov[illegible] will print out 'this is a repeating program' 10 times.

- **init** (the loop counter in this example) is declared as an integer va[illegible]
- *Condition:* The code contained in the curly brackets will repeat as lo[illegible] than or equal to 10. This means it will repeat 10 times.
- *Increment:* this is set so that the loop counter increments by 1 after [illegible]

Using the loop counter

In this program the loop counter is used to indicate in the console window the number of the repetition.

```
this is a repeating p
this is a repeating p
this is a repeating p
this is a repeating p
this is a repeating p
this is a repeating p
this is a repeating p
this is a repeating p
this is a repeating p
this is a repeating p
```

```
class Program
{
    static void Main(string[] args)
    {
        for (int i = 1; i <= 10; i++)
        {
            Console.WriteLine("this is a repeating program - this is
        }

        Console.ReadLine();
    }
}
```

Incrementing in larger steps

In this program, the loop counter starts at 1 and goes up in steps of two so [illegible] less than or equal to 10 then the loop counter will go from 1 to 3 to 5 to 7 an[illegible] Therefore, this loop will execute five times. It is also possible for the loop co[illegible] number and decrement instead of increment.

```
this is a repeating program fo
this is a repeating program fo
this is a repeating program fo
this is a repeating program fo
this is a repeating program fo
```

```
class Prog[illegible]
{
    static void Main(string[] args)
    {
        for (int i = 1; i <= 10; i+=2)
        {
            Console.WriteLine("this is a repeating program for odd numbe
        }

        Console.ReadLine();
    }
```

Coding Task: *Program 4.1 – Ordering numbers*

1. Write a program to output all even numbers between 100 and 1.
   This is some of the output you need to produce to the console window.

```
this is a repeating program for even numbers - this is number 100
this is a repeating program for even numbers - this is number 98
this is a repeating program for even numbers - this is number 96
this is a repeating program for even numbers - this is number 94
this is a repeating program for even numbers - this is number 92
this is a repeating program for even numbers - this is number 90
```

   …

2. Amend your program so that it outputs all odd numbers from 1 to 99.
   This is some of the output you need to produce to the console window.

```
this is a repeating program for odd numbers - this is number 1
this is a repeating program for odd numbers - this is number 3
this is a repeating program for odd numbers - this is number 5
this is a repeating program for odd numbers - this is number 7
this is a repeating program for odd numbers - this is number 9
this is a repeating program for odd numbers - this is number 11
```

   …

Infinite loops

The code below will produce an infinite loop, i.e. a loop that continues to ex
program is closed. Infinite loops are occasionally useful.

```
static void Main(string[] args)
{
    for (; ; )
    {
        Console.WriteLine("this is a repeating program");
    }


}
```

Infinite loops and the break statement
It is possible to break out of an infinite loop using a **break** statement. In the
will stop when `num` is equal to 10. However, in this case it would be easier t
beginning of the loop rather than adding an if break statement inside an infi

```
static void Main(string[] args)
{
    int num = 1;
    for (; ; )
    {
        Console.WriteLine("this is a repeating program");
        num++;
        if (num == 10)
        {
            break;
        }
    }
    Console.ReadLine();
```

Generating random numbers

It is common in programming to generate random numbers, and C# provide
when we need to create a random number. First we declare a new RANDO
specify the range that the number should be generated from using the **.Ne**

Example 1 – generating a random number between 1 and 10

```
int randomnumber;
//first a variable is declared to hold the random number
Random rnd = new Random();
// then declare a new random object
randomnumber = rnd.Next(1, 10);
//then the.Next method of the random object is used to ge
// the number generated will be >= the first number and <
```

Example 2 – generating a random number between 0 and 52
In this case there was only a need to place the maximum value in the bracke

```
int cardnumber;
//first a variable is declared to hold the random number
Random rnd = new Random();
// then declare a new random object
cardnumber = rnd.Next(52);
//then the.Next method of the random object is used to ge
// the number generated will be >= 0 and < than 52
```

Using a loop to generate multiple random numbers
When there is a need to generate multiple random numbers, the **.Next** method is placed inside the loop whereas the declaration of the random object is placed outside the loop.

```
Your ca
Your ca
Your ca
Your ca
Your ca
Your ca
Your ca
Your ca
Your ca
```

```
int cardnumber;
Random rnd = new Random();
// then declare a new random object
for(int i = 1; i < 10; i++)
{
    cardnumber = rnd.Next(1,52);
    Console.WriteLine("Your card number {0} has a value of
}

//the loop will generate 9 cards with 9 different random v
```

Coding Tip

Find out what happens if you place the Random object declaration stateme
(`Random rnd = new Random()`) inside the loop.

Coding Task: *Program 4.2 – Times Table*

Write a program that will output the times table for whatever number the us
The output to the console window should look something like the output wi

```
Which times table do you want to learn ?
2
1 x 2 = 2
2 x 2 = 4
3 x 2 = 6
4 x 2 = 8
5 x 2 = 10
6 x 2 = 12
7 x 2 = 14
8 x 2 = 16
9 x 2 = 18
10 x 2 = 20
11 x 2 = 22
12 x 2 = 24
```


Coding Task: *Program 4.3 – Weight Conversion*

Write a program that displays a conversion table for pounds to kilograms, ra
20 pounds [1 pound = 0.4536 kg]. The output to the console window shoulc
output window below.

```
Conversion: Pounds to Kilograms
Pounds      |      Kilograms
1                  0.4536
2                  0.9072
3                  1.3608
4                  1.8144
5                  2.268
6                  2.7216
7                  3.1752
8                  3.6288
9                  4.0824
10                  4.536
11                  4.9896
12                  5.4432
13                  5.8968
14                  6.3504
15                  6.804
16                  7.2576
17                  7.7112
18                  8.1648
19                  8.6184
20                  9.072
```

The WHILE loop

This loop is a conditional loop. This means that it will terminate only when a evaluates to false. This condition is checked before the loop statements are will continue to iterate (repeat) while the outcome of the condition is true. Th conditional loop are shown below.

```
while (Condition)
{
    Code to be repeated goes here
}
```

A worked example

The program below is part of a larger program, but this snippet is checking on a menu whether the user is either 1 or 2 (as these are the only valid choices). The a choice until it receives a valid choice.

```
bool valid = false;
int choice;


while (valid == false)
{
    Console.WriteLine("please enter your ch
    choice = int.Parse(Console.ReadLine());
    if (choice == 1 || choice == 2)
    {
        valid = true;
    }
}
Console.WriteLine("well done your choice is
```

This condition will be checked b inside the loop execute. If the c (i.e. valid is false), then the stat block execute. If the **condition** the statements inside the loop b and the program jumps to the n loop block

As can be seen from the output from this program (below), the loop only sto enters a valid choice. In this case, the loop executes four times.

```
please enter your choice
3
please enter your choice
8
please enter your choice
9
please enter your choice
2
well done your choice is valid
```

Coding Task: *Program 4.4: The Question Program*

Write a program that asks the user a question, checks the answer and outp
how many tries the user took to get the answer right.

You can ask whatever question you prefer but the output screen below sho
question.

```
What is 2 + 2?
3
Wrong - try again
What is 2 + 2?
4
Well done - you got it right in  2 tries
```


The DO WHILE loop

This loop is similar to the WHILE loop except the code statements inside th
execute at least once. This is because the condition is tested at the end of t
beginning. The syntax and layout for a **`do while`** conditional loop are show

```
do
{
    Code to be repeated goes here
}
while (Condition);
```

A worked example

```
do
{
    Console.WriteLine("please enter your choice")
    choice = int.Parse(Console.ReadLine());
    if (choice == 1 || choice == 2)
    {
        valid = true;
    }
}
while (valid == false);
Console.WriteLine("well done your choice is valid
```

The **do** keyword is placed
the loop, and the **while** k
condition are placed at th

Coding Task: *Program 4.5 – The Basket Program*

Write a program that asks a user whether they want to add items to their ba
price of the item if they say yes. When they have finished adding items the
many items are in the basket and the total price for those items.

The image below shows how the program should look in the output window

```
do you want to put an item in your basket - respond Y for yes an
Y
What is the price of your item?
3.99
do you want to put an item in your basket - respond Y for yes an
Y
What is the price of your item?
2.99
do you want to put an item in your basket - respond Y for yes an
Y
What is the price of your item?
3.00
do you want to put an item in your basket - respond Y for yes an
N
You have 3 items and the value of your basket is 9.98
```

Coding Task: *Program 4.6 – The Password Program*

Prompt the user to enter their password; you then have to check their pass
system. If their password is correct, display 'You have successfully logged i
prompt for a password.

For an extra challenge, if the user enters an incorrect password three time
'Sorry you are out of tries' and quit.

```
please enter your password
GHK
password incorrect - you have 2 tries left
please enter your password
GGG
password incorrect - you have 1 tries left
please enter your password
TRY
password incorrect - you have 0 tries left
sorry you are out of tries
```

Looping consolidation – fixed loops

Program 4.7

Write a maths program that outputs random number multiplication question
then checks the given answers – keeping a score until 10 questions have b
one point for every correctly answered question.

At the beginning of the program, the user should be asked whether they are re
will output questions, check answers and keep score only if they reply true to t
played, then an alternative message, "sorry you didn't want to play", or a simila

At the end of the test a message should be output saying that the test is finishe

The output from the console window should look something like the output

```
 are you re  be  - enter true for yes or false for no
true
what is the  to 6 x 3
18
Well done that is correct, your score is now 1
what is the answer to 5 x 9
45
Well done that is correct, your score is now 2
what is the answer to 2 x 11
22
Well done that is correct, your score is now 3
what is the answer to 2 x 7
14
Well done that is correct, your score is now 4
what is the answer to 7 x 11
77
Well done that is correct, your score is now 5
what is the answer to 8 x 6
48
Well done that is correct, your score is now 6
what is the answer to 10 x 11
110
Well done that is correct, your score is now 7
what is the answer to 10 x 8
80
Well done that is correct, your score is now 8
what is the answer to 9 x 5
45
Well done that is correct, your score is now 9
what is the answer to 11 x 8
88
Well done that is correct, your score is now 10
that is the end of the test, your  final score is 10
```

Evidence required:

1. An **annotated code listing** for the program above.
2. **Screenshots of testing** carried out where:
   a. The user enters 'true' and then answers all the multiplication quest
   b. The user enters 'true' and then answers 8 out of 10 multiplication q
   c. The user enters 'false' and appropriate exit message appears
3. Explain what changes you would need to make to the program for it to
   (addition, multiplication, subtraction, division, modulus).

   .....

   .....

   .....

Looping consolidation – choosing loops

Here you are expected to use the most appropriate loops to solve three sh
This may involve using a for loop, a while loop or a do while loop.

Program 4.8

A credit card has a balance of £50 owing on it. The bank charges 2% interest
that works out how many months it would take for the balance plus interest ch

Evidence to submit for this task program:

1. An annotated code listing for the above program.
2. A screenshot of the output window after the program has been run.
3. An explanation of why the loop you chose was appropriate for this prog

.....

.....

.....

.....

Program 4.9

If we list all the natural numbers below 10 that are multiples of 3 or 5, we ge
these multiples is 23. Write a program to find the sum of all the multiples of

Evidence to submit for this task program:

1. An annotated code listing for the above program.
2. A screenshot of the output window after the program has been run.
3. An explanation of why the loop you chose was appropriate for this prog

.....

.....

.....

.....

Program 4.10

Each new term in the Fibonacci sequence is generated by adding the previ
with 1 and 2, the first 10 terms will be:

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, …

Write a program to find the forty-second Fibonacci number.

Evidence to submit for this task program:

1. An annotated code listing for the above program.
2. A screenshot of the output window after the program has been run.
3. An explanation of why the loop you chose was appropriate for this prog

.....

.....

.....

.....

Chapter 5 – Arrays

What is an array?

An array is a type of data structure. Arrays are more complex than individua
multiple values in indexed locations.

The need for arrays in programming

There is frequently a need when designing programs to store multiple relate
teacher wishes to store the names of the students in a Computer Science c
of arrays, the students' names would have to be stor individual variabl
quite cumbersome when you have a lot of val to stored.

```
string Student1 "Cathy";
string Student2 = "Ben";
string Student3 = "Fred";
string student4 = "Florence";
string student5 = "James";
```

The most efficient method of storing the names is in a **string array**.

How is an array structured?

A one-dimensional array is structured into indices with each index storing a
values can be accessed and changed using the index number. It is commo
starting with 0.

So, our student name example would be structured like the table below:

Array StudentNames

Index [0]	Index [1]	Index [2]	Index
"Cathy"	"Ben"	"Fred"	"Floren

If Cathy leaves the class and is replaced by Sarah, then the appropriate inde
`StudentNames[0] = "Sarah"`

This would **overwrite** the value in index [0] of the names array.

If Cathy leaves the class and is not , then the value in index [0] wo
string like below:
`StudentNames[0] =`

Written Task – Accessing Arrays

An array was created to store the results of a test. Answer the questions be

Array TestMarks

[0]	[1]	[2]	[3]	[4]
66	34	56	13	45

1. What is the value contained in index [6]?
2. How many indices does this array have?
3. What is the index number for the value 45?
4. After a test re[illegible] the value in index [2] is changed to 61. Write a stat

Declaring an array in C#

Arrays are declared like so:

```
string[] StudentNames = new string[5
```

1 2 3

1. **Array datatype** – in this example, the datatype of the index values will other datatypes (mentioned in Chapter 2) could be used. In C# it is not datatypes together in the same array, although it is possible to create a and then create an array of records (see further on in this chapter).
2. **Array identifier** – this is the name we give to the array.
3. **Size** – `new string [5]` creates a new empty string array of five indice

Initialising an array in C#

Like a variable, an array can be declared and initialised at the same time. S

```
string[] StudentNames = new string[] { "Sarah", "Fred", "Mar
```

The values are placed in curly brackets separated by commas at the end of the array. In this case there is no need to state the number of indices to be index for each stated value.

Alternatively, values can be passed into the array either all at once using a array (example 1), a for each loop (example 2) or individually (example 3).

Example 1 – populating an array with user input using a loop

In this example, the for loop is set to iterate one less than the length of the a
element has the value of 0). The length of the array can be found by access
the array object.

```
string[] StudentNames = new string[5];

for (int i = 0; i < StudentNames.Length; i++)
{
    Console.WriteLine("what is the name to be stored");
    StudentNames[i] = Console.ReadLine();
}
```

Example 2 – populating an array using a FOR EACH Loop

A **for each** loop is a special type of loop that can be used specifically for lo
different elements of an array.

```
string[] studentnames = new string[5];
int count = 0;
foreach(string name in studentnames)
{
    Console.WriteLine("What is the name to be stored?");
    studentnames[count] = Console.ReadLine();
    count++;
}
```

Example 3 – populating array indices separately

If rather than populating an array all at once we want to just access a single e
assignment statement. In this example, the fourth element of the array is bein

```
studentnames[4] = "Mary";
```

Accessing an array in C#

Once an array has been declared and initialised, the array can be accessed
for the checking of conditions in iteration and selection statements using the
TIP – use a for loop or a for each loop if you want to access each element i

Coding Task: *Program 5.1*

Create a program that inputs and stores football club names in an array of f
outputs the contents of the array to the screen.

```
What is the name of the football club you want to store
liverpool
What is the name of the football club you want to store
everton
What is the name of the football club you want to store
man united
What is the name of the football club you want to store
stoke city
What is the name of the football club you want to store
man city
liverpool
everton
man united
stoke city
man city
```

Searching and sorting an array in C#

There are two operations which are commonly performed on arrays: search

A simple method of finding out whether a value is in the array is to use a lin is accessed in turn to see if whether is the value being sought. A more effic using the **`.BinarySearch`** method (see below) on an already sorted list.

Arrays can be easily sorted using the **`.Sort`** method (see below).

Coding Task: *Program 5.2*

Adapt program 5.1 so that the program asks for a particular football club an to see whether that club is in the list or not. Use a linear search first (check one you are searching for) and then using a binary search. Remember t must sort the list first.

```
What fo l club do you want to search for
Spurs
That club is NOT in the list
```

Useful array methods and properties

The array object has a number of useful methods and properties that allow done on the array contents.

Method/property	Used for	
`.Length`	Finding the number of elements in the array	**`Clubs`**
`.Sort()`	Sorting the elements into order	**`Array`**
`.Reverse()`	Reverses the original order of the elements	**`Array`**
`.BinarySearch()`	An efficient search to be used on a sorted list. Finds the specified element and returns the index value of the element.	**`Array`** **`clubn`**

Written Task

Find two other methods that can be used on an **`array`**.

Two-dimensional arrays

A two-dimensional (2D) array is used to create a table of data in rows and c
type. An example of this is an array created to show the state of a noughts

Array index	[0]	[1]	[2]
[0]	X	O	X
[1]	X	X	X
[2]	O	X	O

A 2D array would be **declared** like so:

```
char[,] gameboard = new char[3, 3];
```

Initialisin array

```
char[,] gameboard = { { 'X', 'O', 'X' }, {'X','X','X' }, {'O',
```

Accessing each element in a 2D array

```
static void Main(string[] args)
{
    int[,] board = new int[4, 4];
    int counter = 10;

    for (int i = 0; i < 4; i++)
        for (int j = 0; j < 4; j++)
        {
            board[i, j] = counter;
            counter += 2;
        }

}
```

This code is popu
with a set of num
going up by two f

The outer **for** loo
rows in the 2D a
looping through t

Each element in
assignment state

When both loops
look like this:

	[0]	[1]
[0]	10	12
[1]	18	20
[2]	26	28
[3]	34	36

Written Task: *Accessing 2D Arrays*

An array was created to store the results of an exam of two papers. Answe

Array TestMarks

	Student [0]	Student [1]	Student [2]	Student [
Paper [0]	34	56	13	45
Paper [1]	66	91	56	78

1. What is the value contained in TestMarks [1, 1]
2. How many values can be stored in this array?
3. What is the index reference for the value 45?
4. After a remark, the value in index [2,2] is changed to 94. Write a st

Coding Task: *Program 5.3 – Outputting a 2D Array to a Gri*

Investigate how to output the 2D array below, showing the state of a game window so that it outputs a grid with the array elements in each section of th

	[0]	[1]	[2]	[3]
[0]	**"Patrol"**			
[1]				**"Ship"**
[2]				
[3]		**"Destroyer"**		
[4]		**"Destroyer"**		
[5]				

Dynamic arrays

In C# when we want a dynamic array (an array that grows automatically wh
we can use an object called an **`ArrayList`**.

Declaring an `ArrayList`

Firstly **`System.Collections`** needs to be added to the collections above th

```
using System;

using System.Collections;

using System.Collections.Generic;
using System.Linq;
using System.Text
using System.Threading.Tasks;

namespace ConsoleApp2
{
    class Program
```

Then a new **`ArrayList`** can be declared in your program.

```
ArrayList students = new ArrayList();
```

There are a number of useful methods and properties that can be used on t

Method/property	Used for	
`.Count`	Counting up the number of elements in a list. This allows access to the `Count` property of the `ArrayList` object.	nu st
`.Add()`	Adding an element to the end of a list	st
`.Remove()`	Removing a specified element from the list	st
`.RemoveAt()`	Removing an element from a specified index in the list	st
`.Sort()`	Sorts the list in order	st
`.Reverse()`	Reverses the order of the list	st
`.Contains()`	Checks whether a specified value is in the list	st

Written Task

Find two other methods that can be used on an **`ArrayList`**.

Coding Task: *Program 5.4 – Charity Lottery*

Create a program that will help a charity to run its own lottery.

Players can either input their own numbers (five numbers between 1 and 6
generate a lucky-dip set of numbers (five numbers between 1 and 60). If a
contains a multiple of 10 (10, 20, 30, 40, 50, or 60) then they can have an a
number should be randomly generated between 1 and 60).

Once a player's numbers have been input or generated, their line of numbe
and output to the player.

Structs and arrays of records

What is a record?

A record is ection of data items about an object, a person or a thing.

For example, a teacher holds details about her students and their exam ma
column or a **field** and all the items for a particular student are stored in one

name	marks
Lucy	23
Ben	45
David	62
Louise	49

This is one record.

We could consider storing this information in a 2D array, but this is not poss
cannot store multiple data types together in the same array and here we wo
and integer for the marks.

What is a structure?

In C# we refer to a record as a structure (using the keyword **`struct`**). Struc
types and then we can store multiple records in an array (of records).

Declaring a structure for a single record

```
public struct ExamMarks
{
    public string name;
    public int marks;
}
```

This declares the individual valu
data types for a single record. Th
then known as a user-defined da

Declaring a single record object

```
ExamMarks AStudent = new ExamMarks();
```

This ob
single r

Assigning values to a single record object

```
AStudent.name = "Sally";
AStudent.marks = 23;
```

Each value is accesse separately using `objectname.value`

Declaring an array of records

```
ExamMarks[] StudentMarks = new ExamMarks[2];
```

Assigning/accessing values in an array of records

```
StudentMarks[0].name = "Sally";
StudentMarks[0].marks = 23;
StudentMarks[1].name = "Fred";
StudentMarks[1].marks = 45;
```

Each reco using its in the array i

Coding Task: *Program 5.5 – Superheroes Cards*

A toy company is creating an electronic card game for young children that

The following values need to be stored:

- Superhero name (string)
- Lifespan (integer)
- Skill (string)
- Strength (integer)
- Speed (real)

Your program should:

- Store the names of at least four superheroes
- Allow for the input of the superheroes
- Output the input superheroes from the array to the console window

Program 5.6

Load the partially built **Program 5.6.txt** into a new console mode applicatio

At the moment the program contains an empty board array and a procedur
the board array to the console window.

You are to add more functionality to the Fruits program by writing code that

- Create a **`struct`** for the fruit record

Struct FRUIT	Data Type	
FRUITIDENTIFIER	Char	"A"
FRUITNAME	String	"Ap
FRUITVALUE	Integer	10

- Create an **array of seven fruits**
- **Randomly generate coordinates for the fruit identifier** (char) to be p
- Write code that will enable a player to **inspect a fruit** by entering the c
game board

The console window below shows the required output.

```
The board looks like this

   0   1   2   3   4   5   6   7   8   9
0   |   |   |   |   |   |   |   |   |   |
1   |   |   |   |   |   |   |   |   |   |
2   | O |   |   |   |   |   |   |   |   |
3   |   |   |   |   |   |   |   |   |   |
4   |   |   | P | A |   |   |   |   |   |
5   |   |   |   |   |   |   |   |   |   |
6   |   |   |   |   |   |   | R |   |   |
7   |   |   |   |   |   |   |   |   |   |
8   |   |   |   |   |   |   |   |   |   |
9   |   |   |   | G | B |   | S |   |   |
Which Fruit would you like to inspect - enter its row number
9
Which Fruit would you like to inspect - enter its column number
4
the value stored at this location is 'G'
The fruit name is Gooseberry
The fruit value is 12
```

Evidence re ire
1. An **an d code listing** for the program above.
2. **Screenshots of testing** carried out

Extension challenge

Add further code that will total up the fruit values and output a score to the

Chapter 6 – Subprocedures and

The Static Void Main () procedure

In a console mode program, execution of the program will start here. The c
stores a secret word and then allows the user to either guess the word, cha
the program.

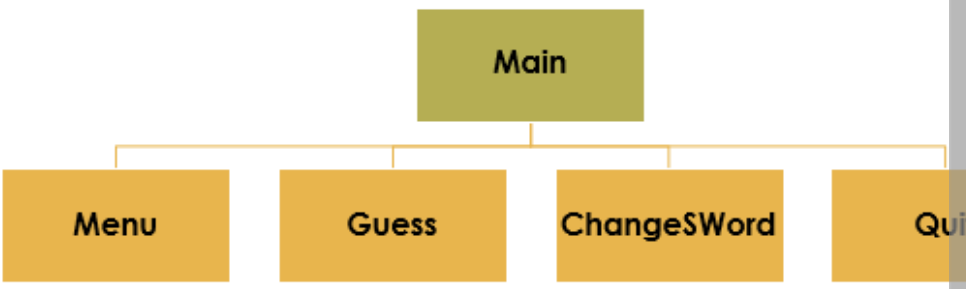
The secret word program

```
class Program
{
    static void Main(string[] args)
    {
        string secretword = "Computer";
        string guess;
        int menuoption;
        do
        {
            Console.WriteLine("Welcome to the guessing program menu - choose
            Console.WriteLine("1 - change the secret word");
            Console.WriteLine("2 - Make a guess");
            Console.WriteLine("3 - Quit");
            menuoption = Convert.ToInt32(Console.ReadLine());

            switch (menuoption)
            {
                case 1:

                    Console.WriteLine("What is the secret word");
                    secretword = Console.ReadLine();
                    break;
                case 2:
                    Console.WriteLine("guess the secret word");
                    guess = Console.ReadLine();
                    if (guess == secretword)
                    {
                        Console.WriteLine("Well done - you have guessed the s
                    }
                    else
                    {
                        Console.WriteLine("sorry that is not the secret word"
                    }
                    break;
                case 3:
                    Console.WriteLine("thank you for playing secret word");
                    Environment.Exit(0);
                    break;
            }
        }
        while (menuoption == 1 || menuoption == 2);

        Cons  Re
    }
```

So far, all the programming we have done (in the other chapters) has place
`Void Main ()` procedure. However, it is more common (and more efficient
down programs into smaller chunks with individual procedures performing s
word program could be decomposed into four separate tasks:

1. outputting the menu
2. making a word guess
3. changing the secret word
4. quitting the program

Top-down diagram to show how the secret word program could



Writing a procedure that does not return a value

A procedure is a block of code that will perform a single task or a set of rela
by another procedure (like the `static void Main ()`).

The proce elow will **output the menu** for the secret word program.

```
static void OutputMenu()
{
    Console.WriteLine("Welcome to the guessing program menu - choose you
    Console.WriteLine("1 - change the secret word");
    Console.WriteLine("2 - Make a guess");
    Console.WriteLine("3 - Quit");

}
```

A procedure
desired ope
value to the
methods ha
case, it can
assignment

NOTE: in C# all procedure identifier names must begin with a **capital letter**.

For this procedure to be executed it needs to be **CALLED** by another proce

How to CALL a procedure

The **`OutputMenu()`** procedure from above is **called** in the **`Main()`** procedu
the **`OutputMenu()`** procedure and then control will be passed back to the M
execution of the **`OutputMenu()`** procedure is complete.

```
class Program
{
    static void Main(string[] ar
    {
        string secr computer";
        st g
        menuoption;
        d
        {
            OutputMenu();
            menuoption = Convert.ToInt32(Console.ReadLine());
```

In order to call the procedure to execute, we need to state the **identifier**
brackets. We will be using the curved brackets to pass **parameters** later

Coding Task: *Program 6.1 – The Secret Word Program*

Copy the code from Program 6.1.txt into a new project. Adjust the program separate procedures for:

- outputting the menu
- making a word guess
- changing the secret word
- quitting the program

Calling procedures with parameters

Sometimes the procedures you call will need to have **values passed** with t can **use** those values to carry out some task. The program below is used t person can be allowed entry into an over-18 venue.

```
static void Main(string[] args)
{
    int age;

    Console.WriteLine("please enter your age");
    age = Convert.ToInt32(Console.ReadLine());
    Entry(age);
    Console.ReadLine();
}

static void Entry(int age)
{
    if (age > 18)
    {
        Console.WriteLine("you are allowed in");
    }
    else
    {
        Console.WriteLine("you are not allowed in as you are u
    }
}
```

In order for the **Entry** whether or not a pers it needs to have the a the procedure as a va **actual parameter or**

When the entry procedure is written, **the formal parameters or arg** curved brackets next to the procedure identifier. `int` refers to the fa must be an integer data type. The identifier of the formal parameter name as the actual parameter.

Coding Task: *Program 6.2 – Adding Numbers Together Us*

Write a program that inputs two numbers and outputs the result of adding t of the two numbers can be done by the `main` procedure. The **addition** and separate procedure called by the `main`.

```
please enter num1
4
please enter num 2
5
the answer is 9
```

What is a function?

A function is simply a procedure that will **return a value** to the calling proce
have written so far have all been **`void`** – this means that they have not retu
procedure after execution.

Program 6.2 called the procedure **`AddNumbers()`** to add two numbers toge
The code below shows this program adapted to use a **function** – instead o
returns the **`answer`** to the calling procedure and then the result is output by
instead. A **function** can be used as part of an **assignment statement** as it
calling procedure.
Every function is a type of procedure or method, but not all procedures are

Functions **always** return a value whereas this is not true of other procedure

An example of the syntax used in functions

```
static void Main(string[] args)
{
    int num1;
    int num2;


    Console.WriteLine("please enter num1");
    num1 = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("please enter num 2");
    num2 = Convert.ToInt32(Console.ReadLine());
    Console.WriteLine("the answer is {0}", Add(num1, num2));
    Console.ReadLine();
}

static int Add(int num1, int num2)
{

    int answer = num1 + num2;
    return answer;

}
```

The func
paramet

As the function r
longer use the v
use the **data typ**

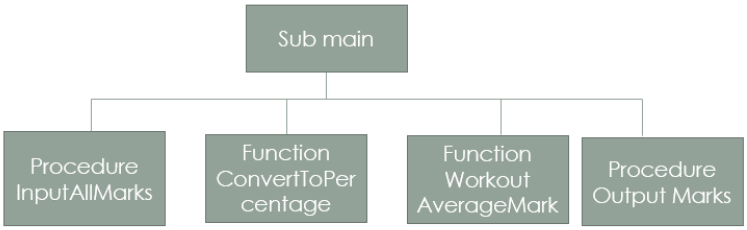
As this is a function it must have a r
This sends the value back to the ca

Coding Task: *Program 6.3 Calculating Percentages Using*

A teacher has set her students a programming exam paper. She has five st
marks for her students are (22, 31, 44, 56, 22). These marks must

Your program needs to work out and display for this paper:

- the percentage value for each student
- the average percentage value for the class (average number)



The ref keyword

By default, parameters are passed to a procedure or function **'By Value'**. T
locations are created for the parameters and, therefore, the value of the ori
changed. By contrast, if we use the keyword '**ref**' in front of a parameter va
are not created and, therefore, we are changing the original argument valu

- **By Value** – a value parameter is a piece of data used by the proced
changes made to it inside the called procedure are not passed back
- **By Ref** – this is used when a piece of data needs to be passed to a
the original value and the changed value passed back to the calling

How to use the ref keyword

```
static void Inputmarks(ref double[] marks, ref double[] p
{
```

The **ref** keyword is used in front of the formal parameter/argument. This m
made to the original array (in its original memory location). It is also placed
parameter/argument in the procedure call.

```
Inputmarks(ref marks, ref percentagemarks);
```

An example of where the ref keyword would be required

```
static void Main(string[] args)
{
    int total = 0;

    AddToTotal(total);
    Console.WriteLine(total);
    Console.ReadLine();
}

static void AddToTotal(int total)
{
    Random rnd = new Random();
    total += rnd.Next(10);
}
```

In this program the **Mai**
variable called **total** a

This value is then pass
parameter. **AddToTota**
number up to 10 and a

When **AddToTotal()** f
procedure will output t

When the program is run

This program outputs 0 to the console window even though an integer of 8 i
number to be added to **total**. The image below shows the value of total af
AddToTotal().

Name	Valu
System.Random.Next returned	8
rnd	{Syst
total	8

Try building this in a console application and using the **Step Into** tool (debug

Then try adding the keyword **ref** to the procedure **AddToMarks(ref int to**
call **AddToMarks(ref total)**. This time the new value generated by **AddT**
console window.

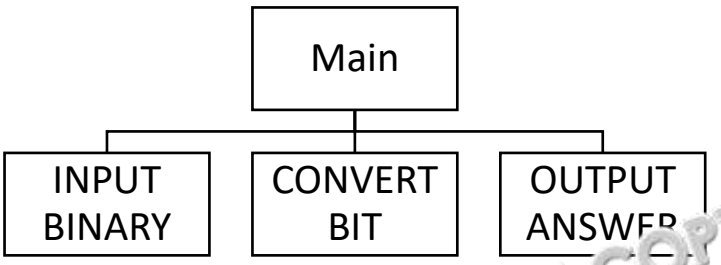
Program 6.4

The following algorithm describes the process by which an 8-bit binary numb
equivalent. The top-down design diagram shows how the algorithm could be

Program the algorithm in C# by selecting appropriate structures and statem
procedures and functions to represent the different tasks.

Algorithm to convert an 8-bit binary number to denary
Answer = 0
Column = 128
OUTPUT “Please enter the binary n to be converted”
Binarynumber = USERIN
FOR I = 0 Bin ber.length - 1
Bitvalue =
IF binarynumber(i) == “1” THEN
bitvalue += column
ELSE
Bitvalue = bitvalue
END IF
Column /= 2
Answer += bitvalue
END FOR
OUTPUT Answer

This is the suggested task breakdown for the converter program. **`Convert`**B
every bit in the binary number.



Evidence required:

1. **Provide your t e sting** for the above program.
2. **Test y rogram** with the following binary numbers:
   a. 10 0000
   b. 11111111
   c. 10010011
3. **Adapt the program** so that it would work for any length of binary numb
   a. **Provide your adapted code** for the above.
   b. **Test** with the following binary numbers:
      i. 0011111110
      ii. 1111
      iii. 001101

Chapter 7 – String Handl

Strings and string handling in programming

- A **string** is a collection of characters and is treated in C# as an object.
  array of characters.
- A **character** is anything on a computer keyboard and is an elementary

It is frequently necessary as part of standard computer processing to
C# allocates 2 bytes per character using Unicode – a coding system desig
The first 256 characters of Unicode are the same as ASCII:

- Upper-case letters are stored as numbers 65–9
- Lower-case letters are stored as numbers 97–122
- Numeric digits 0 to 9 are stored as numbers 48–57
- A space character is stored as number 32

Commo rations you might perform on strings

What processes might involve looking at the individual characters within a s

- Finding the **length** of the string (**`string.Length`**)

  It can be useful to know the length of a string in case you want to loop
  elements, in which case you can use the string length to set the **for-l**

```
string s = "This is some text";
Console.WriteLine(s.Length);
//this will output the number 17
//17 is the number of characters including spaces in s
```

- Finding the **index** of the first occurrence of a value within a string (**str**

  This is essentially finding out at what index point in the string a specifie
  string function will return a numerical value that is the index position nu
  character in the string.

```
string s = "This is some text";
Console.WriteLine(s.IndexOf('s'));
//this will output the number 3
//the 3rd index in the string is the first occurence o
//NOTE: numbering of the indexes starts at 0
```

- **Inserting a value** into a string at a particular point (**`string.Insert(In**

```
string s = "This is some text";
s = s.Insert(12, "more ");
Console.WriteLine(s);
//this will output "This is some more text"
//the string "more " has been inserted into string s
```

- **Removing characters** from a specified index onwards (**`string.Remov**

```
string s = "This is some text";
s = s.Remove(12);
Console.WriteLine(s);
//this will output "This is some"
//all characters after the 11th index have been remov
```

- **Replace all instances of a specified value** in a string (`string.Replace(`

```
string s = "I am a student";
s = s.Replace("student", "policeman");
Console.WriteLine(s);
//this will output "I am a policeman"
//The string "student" is replaced with the string"polic
```

- **Return a substring of characters** starting at a particular **index** (`string.S`

  The first parameter is the starting position of the new string. The secon number of characters that will be extracted.

```
string s = "This is a string";
s = s.Substring(10, 6);
Console.WriteLine(s);
//this will output "string"
```

- **Returns a true or false value** if a string contains a particular value (`st`

```
string s = "This is a string";
bool found;
found = s.Contains("string");
Console.WriteLine(found);
//this will output true
```

- **Splits a string** (`string.Split('character')`)

```
string s = "This is a string";
char delimiter = ' ';
string[] substrings = s.Split(delimiter);
foreach ( var word in substrings)
{
    Console.WriteLine(word);
}
```

  In this example the string is being split where there is a space (an emp

Research Task: ***Other String Methods in C#***

The examples above show some of the most common and useful string-ha

Do some research to find out about **at least two** other methods that can be

Make a no[illegible] what they do and what syntax is required.

Accessing individual characters in a string

You can also access characters in a string by its index – just like you would an array:

```
string s = "This is a string";
Console.WriteLine(s[0]);
//outputs the letter 'T'
```

Processing strings with relational operators

It is possible to use **relational operators** with strings. These operators wor Unicode number.

In the example below, the value of the lower-case letter 't' is 116 and the va is 103. It is these ASCII values that are compared in the selection condition

```
string s = "this is a string";
if (s[0] > s[s.Length - 1])
{
    Console.WriteLine("{0} is higher up the alphabet", s[0
}
else
{
    Console.WriteLine("{0} is higher up the alphabet", s[s
}
;
//the ASCII value for t is 116 and the value for g is 103
//the program outputs "t is higher up the alphabet"
```

Coding Task: *Program 7.1 – The Cat String Program*

Open the **Program 7.1.txt file**, copy it into a new console mode application manipulate the string and output amended strings.

```
static void Main(string[] args)
{
    string cat = "The domestic cat is a small, typically furry, carnivorous ma
    //1. replace all instances of cat with dog and cats with dogs - re-output
    //2. find the length of the string - output the length
    //3. Return the substring "they are often called house cats" and output th
    //4. output the index of the first example of "cat"
    //5. Insert the sentence "There are a number of different breeds." after
    //6. Reverse the first sentence and output it
    //7. Split the string into an array of substrings
    Console.WriteLine("The original string is -  {0}", cat);

    Console.ReadLine();
```

Coding Task: *Program 7.2 – The Space Wars Name Progra*

You are to write a program that comes up with your space wars name by e
input values and concatenating them.

For your first spacename you need to:

- take the first three letters of your real last name
- add the first two letters of your real first name

For your last spacename you need to:

- take the first two letters of your mother's first name
- add the first three letters of the city in which you were born

The screenshot below shows typical output to the console window.

```
Hello and welcome to the Space Wars Name program
Please enter your first name
fred
Please enter your last name
salmon
Please enter your mother's first name
debbie
Please enter the name of the city/town you were born in
manchester
Your space name is salfr deman
```

Working with strings using ASCII values

There are a number of different operations using the ASCII values of chara
when programming with strings.

Assigning a character based on the ASCII value

Here the variable **`letter`** is being assigned the character '**b**' using its ASCI

```
char letter;
letter = (char)98;
Console.WriteLine(letter);
Console.ReadLine();
```

```
b
```

Finding the ASCII value of a character

This program takes in a character, finds its ASCII value and then outputs th

```
int lettervalue;
char letter;
Console.WriteLine("please enter the letter you want a value
letter = Convert.ToChar(Console.ReadLine());
lettervalue = (int)letter;

Console.WriteLine("The ASCII value of {0} is {1}", letter,
Console.ReadLine();
```

```
please enter the letter you want a value
G
The ASCII value of G is 71
```

Finding the ASCII values in a string

This program takes in a string, works out the ASCII value of each character

```
string value = "9quali52ty3";

// Convert the string into a byte[].
byte[] asciiBytes = Encoding.ASCII.GetBytes(value);
foreach (var asciiByte in asciiBytes)
    Console.Write("{0} | ", asciiByte);
Console.ReadLine();
```

```
57 | 113 | 117 | 97 | 108 | 105 | 53 | 50 | 116
```

Coding Task: Program 7.3 – *Password Generator*

Write a password generator program that outputs an **eight-digit** password u

1. Generate a **random number** (between 33 and 127)
2. Use that number to generate a **character**
3. Add that character to a **password string**
4. Check the password string to see whether there is a **capital letter** – if t score for each letter (ASCII values for capitals are 65 to 90)
5. Check the password string for any of the **non-numeric or non-alphab** each one found
6. **Output the password and its strength** – if the score is 40 or more, th 25 or more, it is medium strength; if it is < 25 then it is weak.

The image below shows a typical output.

```
Your password is Qzdu~mq2
your password score is 20
your password is weak
```

Program 7.6

You are to complete a string-checking program in a C# console mode appli

Complete a program for string checking that will:

- Feed in a string containing the sentence 'I am going to check every wo keywords'
- Split the string into an array of the individual words making up the sent
- Check the array of words for instances of the keywords (check, word, s
- Output the sentence with the keywords highlighted (in a different consc

The image below shows typical expected output.

```
I am going to check every word of this sentenc
```

Evidence required:

1. An **annotated code listing** for the above program.
2. A **testing screenshot** showing the required output.

TIP! Changing the colour of the console window and

This statement allows you to change the **console window background co**

```
Console.BackgroundColor = ConsoleCol
```

This statement allows you to change the colour of the **text**:

```
Console.ForegroundColor = ConsoleCol
```

Chapter 8 – Validation and Excepti

The need for validation

Inaccurate data entry can cause a number of issues for organisations and c

For example:

- inaccurate data cannot be relied on by organisations
- inaccurate data may bring an organisation into conflict with data protec
- data entry mistakes may cause programs to crash

VALIDATION is a software check that ensures that dat entered into the pr
are different checks that can be made as follow

- **Checking the length of a** – data entry could be restricted to a c
  For example, when in a telephone number, we can program a c
  numeric input.

```
string telephonenumber;
do
{
    Console.WriteLine("Please enter your phone number");
    telephonenumber = Console.ReadLine();
    if(telephonenumber.Length != 11)
    {
        Console.WriteLine("incorrect length please re-enter
    }
}
while (telephonenumber.Length != 11);
```

- Checking that **data entry is in a particular format** (e.g. lower-case let

```
string name;
int lettervalue;
bool valid;
do
{
    Console.WriteLine("Please enter your name");
    name = Console.ReadLine();
    lettervalue = (int)name[0];
    if(lettervalue >= 97 && lettervalue <
    {
        Console.WriteLine("y ust have a capital lett
        valid = false;
    }
    e
    {
        valid = true;
    }
}
while (valid == false);
```

More sophisticated pattern matching is best done with a regular express

- Checking that **data entered is within a specified range**

```
double price;
do
{
    Console.WriteLine("Please enter the price of the prod
    price = double.Parse(Console.ReadLine());
    if(price < 10 || price > 100)
    {
        Console.WriteLine("incorrect price please re-ente
    }
}
while (price < 10 || price > 100);
```

- Checking that **some data is entered when required** for those instanc
enter anything

```
static bool PresenceCheck( ref string surname, ref bool vali
{

    if (surname == "")
    {
        Console.WriteLine("you must enter a surname");
        valid = false;
    }
    else if(surname != "")
    {
        valid = true;
    }
    return valid;
}
```

- Checking that data is of the **correct expected type.** On the following p
`catch` statement can be used to trap invalid data type entries, so they

Validation using exception handling

Exception handling is the process of dealing with events that might cause a

For example:

- trying to read a non-existent file
- trying to convert a non-numeric string entered by the user to an integer
- entering nothing when an input is expected
- trying to perform calculations with a non-numeric variable
- division by 0

In the exa below, the attempted assignment of a character to an integ
casting exception.

```
Dave's Electronic Address Book - do you want to 1 [e
2 [view address book]
3 [Quit]
a
```

```
Console.WriteLine("2 [view address book]");
Console.WriteLine("3 [Quit]");
menuchoice = int.Parse(Console.ReadLine());
```

Exception Unhandled

System.FormatException: 'In

Using a try…catch block

```
int menuchoice;
bool valid = false;
do
{
    Console.WriteLine("Dave's Electronic Address Book - do you want to
    Console.WriteLine("2 [view address book]");
    Console.WriteLine("3 [Quit]");
    try
    {
        menuchoice = int.Parse(Console.ReadLine());
        valid = true;
    }
    catch (Exception)
    {
        Console.WriteLine("an error has ocurred - please re-enter");
        valid = false;
    }
}
while (valid == false);
```

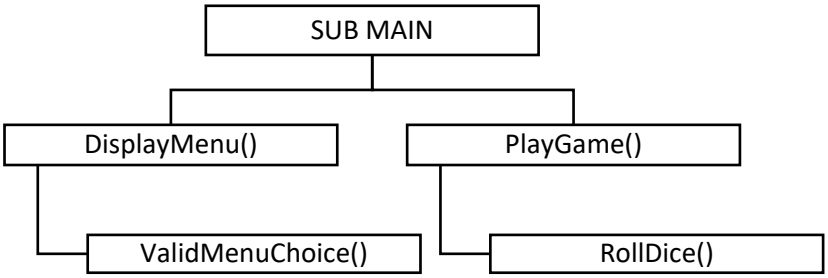
The **try** block tries to execute the li

Coding Task: *Program 8.1 – The Dice Program*

Write a program in C# that will simulate the rolling of a dice.

1. A player plays against the computer until one of them reaches the scor
2. They should then have the option of playing the game again.

The diagram below shows the top-down design for this program.



The procedure `Main()` should:

- call `DisplayMenu()`
- call `PlayGame()` or quit the program, depending on the choice the user
- continue to offer the menu until quit is chosen

The procedure `Menuchoice()` should:

- contain code that outputs the main menu
- have a **`try catch`** and other validation that prevents an inva
- call `ValidMenuChoice()` which makes sure that only option 1 (play a g program) is accepted by the program

The procedure `PlayGame()` should:

- decide who goes first (computer or player)
- call `RollDice()` to return a number which should be added to the score
- check whether the player or the computer is a winner
- change whose turn it is before calling `RollDice()` again
- output who the winner is with an appropriate message

The procedure `RollDice()` should:

- return a random number between 1 and 6 to `PlayGame()`

Program 8.2

Please open the text file called **Program 8.2.txt**

At the moment, the game works in this way:

- It randomly places six items of treasure on the game board.
- When the game is output to the console window, the treasure is hidden

Extend this program so that:

1. It accepts input coordinates from the player so that the player can move the treasure
2. The input of the move coordinates needs to be validated to ensure that coordinates
3. The move coordinates need exception handling to ensure that the program reasonably (entering a character or a string – not entering anything).
4. If the player lands on a place on the grid that has treasure, 10 points are
5. A count will need to be kept of the number of player moves.
6. A player wins if all the treasure is collected and the moves are fewer than computer wins.

Evidence required:

1. An **annotated code listing** for the program above.
2. **Screenshots of testing** carried out

Chapter 9 – Text File Hand

Once a program has finished executing, any data in the variables and data
necessary in programs to retain data for future use. In this case, data which
execution of a program needs to be saved in an external file.

This chapter deals with saving data in text files. There are a number of differen
handling in C#, and here we will look at basic File methods and `StreamReader`

A text file stores all its data as characters represented by their ASCII codes
to be both written to and read from at the same time. A text file is sequentia
beginning of the file to the end.

Using file-handling methods

It is necessary to add `System.IO` to the `using` part of the code module. Se

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
```

Writing *to* a file

To write to a file, the **`WriteAllText()`** method is used. In order to use this
file path / filename of the file we want to read to and the data we want to ad

```
static void Main(string[] args)
{
    string name;
    string surname;
    string filename = "addressBook.txt";

    Console.WriteLine("please enter your fi
    name = Console.ReadLine();
    Console.WriteLine("please enter your su
    surname = Console.ReadLine();
    File.WriteAllText(filename, (name + sur
}
```

Here the
a string

The **`WriteAllText()`** method will create a file with the specified file path / f
to it. If we do not specify the file path and only specify the filename then a fi
by default in the **bin/debug folder** of your C# project. If the file already exis
overwritten.

The program above creates a file called **addressbook.txt** in the bin/debug
input name and surname into it.

Reading *from* a file

```
text = File.ReadAllText(filename);
Console.WriteLine(text);
```

The code above will read all the text from a file and output to a console window. V
method, we pass the filename and possibly the file path as a parameter to this m
path for the file, it will look for the file in the bin/debug folder of the visual studio p

Other file-handling methods

There is a variety of other useful methods that can be used with text files.

Method	Description
`.AppendAllText()`	Appends text to the end of a file
`.Create()`	Creates a file in the specified location
`.Delete()`	Deletes the specified file
`.Exists()`	Determines whether the specified file exi
`.Copy()`	Copies a file to a new location
`.Move()`	Moves a specified file to a new location

Coding Task: *Program 9.1 – Storing usernames and passv*

Write a short program that:

1. Asks for and inputs a username.
2. Asks for and inputs a password.
3. Stores the username and password in a text file.
4. Outputs the contents of the file.
5. Allows for multiple entries.

Try storing more than one entry – what happens to the file?

```
This is a password entry p
1 - add a new username and
2 - Output the username an
3 - Quit the program
```

Writing single entries to text files using StreamWriter

The `StreamWriter` and `StreamReader` classes are alternatives to the file han
above, and there is some memory saving for large files as it writes to files line
line by line. You will still need to import the System.IO namespace in the usi

```
string name;
string surname;
string filename = "address.txt";

Console.WriteLine("please enter your first name");
name = Console.ReadLine();
Console.WriteLine("please enter your surname");
surname = Console.ReadLine();

StreamWriter SW = new StreamWriter(filename);
SW.WriteLine(name + " " + surname);
SW.Close();
```

When
a nev
passi
filena

The
a sing
text t
param

Writing multiple entries to a file

When writing multiple entries to a file it is more efficient to use a `using` stru
stream open so multiple entries can be written to the file. The `StreamWrite`
`using` keyword) will be disposed of when the `using` block has completed e
stream is closed when the block finishes executing.

In the example below, the contents of the array are written to the text file wi
line. If we wanted all items on the same line in the text file, we would just us

```
string[] names = new string[] { "Fred", "Bob", "Mary", "Na
string filename = "Names.txt";

using (StreamWriter SW = new StreamWriter(filename, true))
{
    foreach (var name in names)
    {
        SW.WriteLine(name);
    }
}
```

Reading from files using StreamReader

When reading from a file we can use the **`Peek`** method to decide whether w
the file. The peek method returns an integer value which is the ASCII value
line to be read. When it reaches the end of the file, the integer value returne
end of the file has been reached.

```
StreamReader SR = new StreamReader(filename);
while (SR.Peek() > -1)
{
    Console.WriteLine(SR.ReadLine());
}
SR.Close();

Console.ReadLine();
```

The WHILE loop
the file until the e

Coding Task: *Program 9.2 - Amending the password prog*

Change your passwo g am so that it uses `StreamWriter` and `Strea`
read from t xt .

Try using b ingle entry and multiple entry methods for writing to the text
differences in how the program behaves?

Program simple fantasy football team

Program 9.3

Write a program that allows users to create and store footballers for a fanta
following characteristics:

- name
- goals scored
- number of yellow cards
- number of red cards

When the program loads, a menu sh l b utput asking whether the
team member (no more than fi allowed), view their existing team,
or quit the program (see e ls output below).

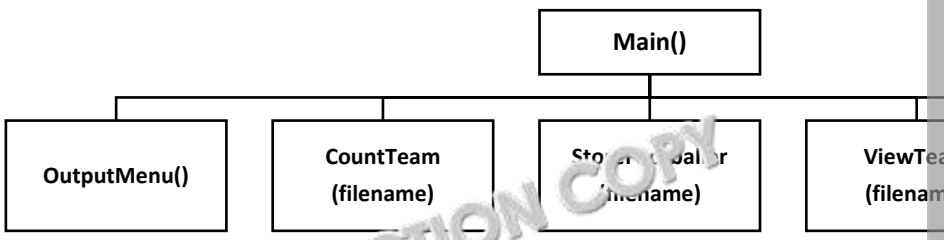
```
Welcome our fantasy team - do you want to:
1: Add a ew player
2: View the team
3: Calculate the team's value
4: Quit the program
```

The team's current value is calculated by adding all the goals scored – 1
value is reduced by red and yellow cards (a 5-point reduction for a red card
yellow card).

The user's team (containing the stats for each player) should be writte
user wants to view their team, the team needs to be read from their file and
window. When the user wants to calculate the current value, the team is rea
values used to calculate the current value of the team.

NOTE: for the purposes of this exercise it is not a requirement to upda

A suitable program structure is shown below:



Evidence re

1. Annota ode for the program

2. Testing to show:
   a. a screenshot of the console window to show five players added
   b. a screenshot to show that trying to add a sixth player results in a wa
   c. a screenshot of the text file to show that five players have been ad
   d. output from the console window to show that the whole team is ou
   e. output from the console window to show that the team value has b

Chapter 10 – Using Class

The event-driven nature of windows applications has led to the developmen
languages.

An object **class** is a grouping together of data structures and behaviours (a

Class attributes

Attributes are the **data items** that are needed for an object of that class. Fo
Science teacher has students in her class. The student object has certain a
Computer Science classroom (see table below).

Class ComputerStudent
Firstname
Surname
FormGrou
TargetGrade
ProgressGrade

How to write a class in C# with just attributes

```
namespace classes_chapter_10
{
    class Program
    {
        static void Main(string[] args)
        {
        }
    }

    class ComputerStudent
    {
        public string firstname;
        public string surname;
        public string form;
        public int targetgrade;
        public int progressgrade;

    }
}
```

The **ComputerS**
outside of **clas**
this class we hav
have been defin
allows these attr
from any part of

How to lass objects in our main program

Creating a new class object

This process of creating a new class object is known as **instantiation**. The
instantiate a new `ComputerStudent` object in C#.

```
ComputerStudent AStudent = new ComputerS
```

Class name | **Object identifier name** | Creates a new o constructor meth

Assigning values to the public attributes

```
static void Main(string[] args)
{
    ComputerStudent AStudent = new ComputerStudent();
    AStudent.firstname = "Tom";
    AStudent.surname = "Wilson";
    AStudent.form = "10H";
    AStudent.targetgrade = 7;
    AStudent.progressgrade = 6;

}
```

There are **two way**
attributes can be a

We can assign val
class_attribute
s shown here, or
below which puts t
within a **code bloc**
class object) and, t
object identifier to

OR

```
class Pr
{
    static void Main(string[] args)
    {
        ComputerStudent AStudent = new ComputerStudent
        {
            firstname = "Tom",
            surname = "Wilson",
            form = "10H",
            targetgrade = 7,
            progressgrade = 6
        };

    }


}
```

It is far more likely, however, that we would write classes with attributes **an**
define the ways in which programs can interact with the class attributes. W
and the methods public.

Coding Task

Try building the program ch that defines the student class and creates a
What happens wh hange the attributes from public to private?

Writing classes with attributes and methods

When attributes are private, the only way programs can interact with an obj
methods we define.

The constructor method

```
class ComputerStudent
{
    private string firstname;
    private string surname;
    private string form;
    private int targetgrade;
    private int progressgrade;

    public ComputerStudent()
    {
        Console.WriteLine("Please enter the firstname");
        firstname = Console.ReadLine();
        Console.WriteLine("Please enter the surname");
        surname = Console.ReadLine();
        Console.WriteLine("Please enter the form");
        form = Console.ReadLine();
        Console.WriteLine("Please enter the target grade");
        targetgrade = int.Parse(Console.ReadLine());

    }

}
```

public C
known as
This is the
when we i
object.

Accessor methods (Get methods)

These methods are used to return the value of a class attribute. Now that o
only way they can be accessed is through methods we provide for this purp

```
public string GetStudentName()
{
    string studentname;
    studentname = firstname + " " + surname;
    return studentname;
}
```

Th
the
at
ob
wi

In our **main program** we would **call** this method like so:

```
static void Main(string[] args)
{
    ComputerStudent AStudent = new ComputerStudent();

    Console.WriteLine("The student's name is {0}", AStudent.GetSt
```

```
Please enter the firstname
Aneka
Please enter the surname
Brown
Please enter the form
10G
Please enter the target grade
4
The student's name is Aneka Brown
```

When the program is run, fir
called which sets the values

Then the **GetStudentName(**
created student object, and t
the console window.

Mutator methods (Set methods)

These methods are provided so that the values of the class attributes can b
the values or only some of them). In the case of the student class example
form or target, grade but we are unlikely to need to change the first name o

```
public void SetProgressGrade(int newgrade)
{
    progressgrade = newgrade;
}

public void SetTargetGrade(int newtarget)
{
    targetgrade = newtarget;
}
```

The ne
new ta
param
and its
approp

Combine essor (Get) and mutator (Set) methods

It is also possible to combine the setting of a class attribute with returning th
method is called in the main program.

```
public int CalcNewTarget()
{
    if (progressgrade > targetgrade)
    {
        targetgrade = progressgrade + 1;
    }
    else
    {
        Console.WriteLine("There is no need to change your targe
    }
    return targetgrade;
}
```

This class method is
a **mutator**. It will ch
targetgrade if the
is more than the ta
returns the newly ca
to the main program

The main program code

```
if (menuchoice == 1)
{

    Console.WriteLine("Hello {0}", AStudent.GetStudentName());
    Console.WriteLine("Your target grade is {0}", AStudent.Cal
}
```

If change a target grade is acc
the CalcNewTarget() method o
for the class object AStudent.

This results in the following output

Where a student target grade is 4 and their progress grade is 5

```
Hello JANET Brown
Your target grade is 6
```

Where a student target grade is 6 and their progress grade is 5

```
There is no need to change your target gr
Hello JANET Brown, your target grade is 6
```

Coding Task: *Program 10.1 – The Cat Program*

The Cats Protection Trust is a charity that takes in abandoned cats. It wants
the cats in the home.

Write a class for the cats that could be used in the charity's management sy
need to store are the cat's name, breed, gender, weight, whether or not it has
vaccinated and whether it is ready for rehoming.

The system will need methods to retrieve all the cat's details and methods th
neutering status, vaccination status and rehoming status to be changed.

Try storing the cats in an array. Class objects can be stored and accessed in
to structs (see Chapter 5, section – 'Structs and arrays of records').

Class Cat
- catName: string
- catBreed: string
- catGender: char
- catWeight: double
- neutered: bool
- vaccinated: bool
- rehomeReady: bool
Methods
+Cat() *sets up a new cat's details*
+ViewCatDetails() *outputs to the console window all the cat's current infor*
+GetCatName() *returns the cat's name*
+ChangeWeight(newweight) *changes the cat's weight for the value passed*
+ChangeVaccStatus(newstatus) *changes the cat's vaccination status to the va*
+ChangeNeutered(newneutered) *changes the cat's neutering status to the val*
+ChangeReHome(rehome) *changes the cat's rehoming status for the valu*

The main menu for the program might look like this:

```
welcome to the Cat's Protection Trust management system. Do y
1 - Add a new Cat
2 - Retrieve a Cat's details
3 - Update a Cat's details
4 - Quit the program
```

The menu for updating a cat's details might look like this:

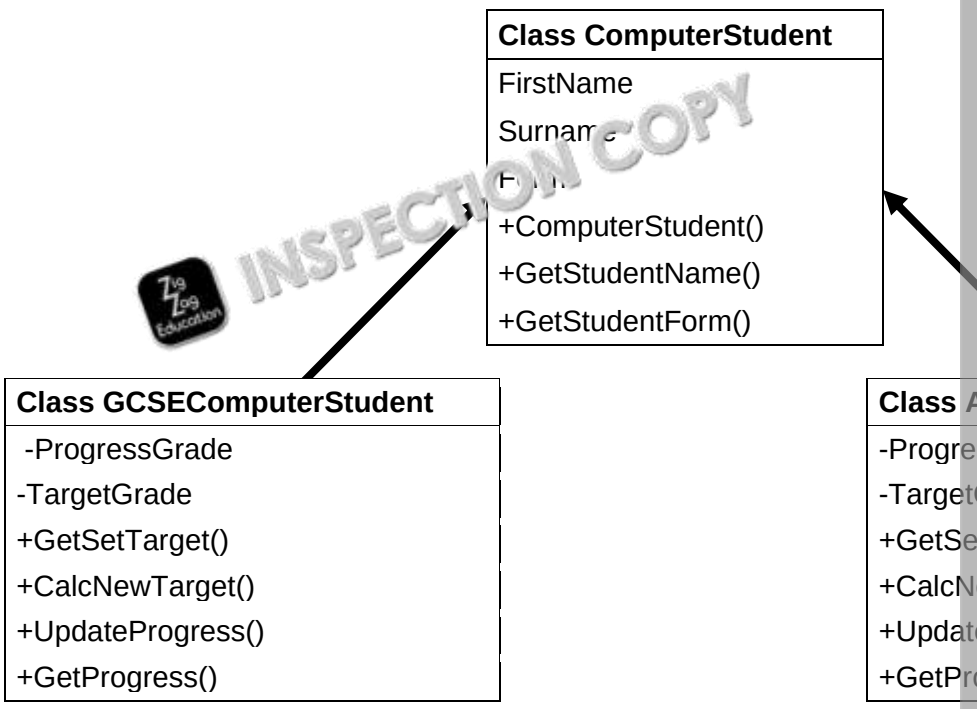
```
What detail would you like to update?
1 - update cat weight
2 - update cat vaccination status
3 - update cat neutering status
4 - update cat rehoming status
5 - Return to main menu
```

Inheritance

You can often define a new type of object by amending the definition of some programming this has the advantage of allowing the reuse of existing code an

Using inheritance

Computer students come in many forms as there are many types of course systems. So, we can adapt the **`ComputerStudent`** class by creating subcla (A Level and GCSE). A Level uses the grading system A–U, and GCSE us



Writing an inherited class

```
class GCSEComputerStudent: ComputerStudent
{
```

When defining our inherited class, we put the name of the inherited class fo the name of the class it inherits from.

Then we decide what additional attributes this type of object needs. (Reme the attributes from the parent class; in this example, class `ComputerStude` `GCSEComputerStudent` class will have `ProgressGra` and `TargetGra`

Amending the scope of the pare class attributes

```
class ComputerStu
{
    pro d string firstname;
    protected string surname;
    protected string form;
```

When we want a subclass to inherit attributes and methods from a class, w of the parent class attributes and any private methods from private to **prot**

The **protected** keyword means that only inherited classes can access thes attribute or method using the keyword protected can be inherited. It is worth may be occasions when it is not desirable to enable all methods and attribu them as private prevents them from being inherited by any subclasses.

Creating and using an inherited class object

How the class methods are called in the main program

Calls parent class (ComputerStudent) constructor met
then constructor method of GCSEComputerStudent cl

```
GCSEComputerStudent AStudent = new GCSEComputerStudent();
int menuchoice;
Console.WriteLine("The student's name is {0}", AStudent.GetS
do
{
    Console.WriteLine("D[illegible] to:");
    Console.Write[illegible] Calculate a new target grade");
    Cons[illegible]Wr[illegible]ne("2 - Input the new progress grade:");
    Con[illegible]WriteLine("3 - View the student's grades");
    Cons[illegible].WriteLine("4 - Quit the grades program");
    menuchoice = int.Parse(Console.ReadLine());
```

And then if the third selection is chosen from the menu:

```
case 3:
    Console.WriteLine("Hello {0}", AStudent.GetStudentName()
    Console.WriteLine("Your target grade is {0}", AStudent.G
    Console.WriteLine("Your current progress grade is {0}",
    break;
```

GetStudentName() is called from the par
(ComputerStudent class) then GetTarge
SetTarget() from the GCSEComputerStu

How the output would look

```
Please enter the firstname
John
Please enter the surname
Smith
Please enter the form
11H
What target do you want to set for this student? Input a grade b
3
The student's name is John Smith
Do you wa[illegible]:
1 - Calcu[illegible] new target grade
2 - Input [illegible] new progress grade:
3 - View the student's grades
4 - Quit the grades program
3
Hello John Smith
Your target grade is 3
Your current progress grade is 0
```

This output and input comes from th
ComputerStudent constructor meth
(Public ComputerStudent()).

This output and input comes from
constructor method of the
GCSEComputerStudent class.

This grade output comes from call
GetStudentName() method of the
followed by a call to GetTarget()
methods of the **GCSEComputerS**

Coding Task: *Program 10.2 – The Grading Program*

Add an A Level student class to ***Program 10.2.***, the Computer Students' G
found in the accompanying text files. A Level students' grades range from A

Class ALevelComputerStudent	***Explanation***
-ProgressGrade	*Holds the current progress grade*
-TargetGrade	*Holds the current target grade*
+ALevelComputerStudent()	*Asks for the target grade to be set a is an acceptable one (A–U)*
+GetTarget()	*Returns the currently set target*
+CalcNewTarget()	*Takes the current target and checks grade is higher. Resets the target to progress grade if it is.*
+UpdateProgress()	*Sets or updates the current progress*
+GetProgress()	*Returns the current progress grade*

Change the main program so that, in addition to its current functionality, it a
you want to create: A Level or GCSE.

Polymorphism and overriding

Polymorphism means 'having many forms'. It occurs when you have classe
which have inherited methods from the base or parent class. A call to an o
different implementation depending on the type of object that calls the meth

Example:
A set of classes has been built for a series of talking toys. Class `SheriffWo`
`BuzzLightYear` inherit from class `TalkingToy`. All the classes have a **`Play`**

However, this method, which uses the same name in all classes, has differe
depending on the object that calls it. These methods are said to be **polymo**

Class TalkingToy
Protected string toyname
Protected string toytype
Public TalkingToy()
Public virtual void PlayMessage()

```
public virtual void PlayMessage()
{
    Console.WriteLine("Hello I am an {0}", t

}
```

Class SheriffWoody
Public SheriffWoody()
Public override void PlayMessage()

C
Public Bu
Public ov

```
public override void PlayMessage()
{
    base.PlayMessage();
    Console.WriteLine("My name is {0}", toyname);
    Console.WriteLine("My most famous saying is");
    Console.WriteLine("Sergeant, establish a recon post downstairs");
    Console.Write("Code Red, You know what to do !");
}
```

The **base** keyword is used to call the virtual method from the parent or base class. If this is not included, the parent or base class method will be ignored.

```
public override vo
{
    base.PlayMess
    Console.Write
    Console.Write
    Console.Write
}
```

The **over** the subc override

```
Hello I am an action fi
My name is Sheriff
My most f u ng is
Sergeant tablish a recon post downstairs
Code Red, You know what to do !
```

```
Hello
My nam
My mos
To inf
```

Coding Task: *10.3 – Talking toys program*

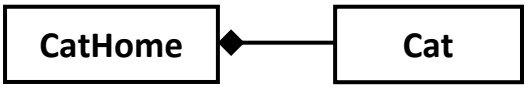
Implement the talking toys class program fully using the definitions on the p
the program asks what toy you want and then outputs your chosen toy char
message it plays.

For example, the output for the Woody toy being chosen would look like this:

```
Welcome to the
Which talking t
1 - Buzz LightY
2 - Sheriff Woo
2
What is the typ
Action figure
Hello I am an A
My name is Sher
My most famous
Sergeant, estab
Code Red, You k
```

Association

Associatio[illegible]different kind of relationship from inheritance. In associatio
within other[illegible]s objects.



In this case, the CatHome object will be a container for the Cat objects. There

Association aggregation
In the case of aggregation by association, the Cat objects contained within the CatHome object are not destroyed when the CatHome object disappears. They can exist in the program independently of the container object.

Composition aggregation
In the case of aggregation by composition, the Cat objects do not exist independently of the CatHome object and are destroyed when the CatHome object is destroyed.

Coding Task: *Program 10.4 – The Cat Program (using agg*

Using the solution for program 10.1 (either your solution or the solution provi
so that it now includes a `CatHome` class that contains the Cat class objects. T
class remains the same as in program 10.1, and the definition for the new Ca
below. The program should work in the same way as before but using the co

Class CatHome
- maxnoofcatsinhome: integer *(set as a constant = 10)*
- CatList [maxnoofcatsinhome]: Cat Array *(this will be the array of cat o*
- Catcount: integer *(this will hold the number of cats currently in the hon*
Methods
+CatHome() *sets the cat count at 0*
+AddCat() *adds a new cat to the cat array if there is space*
+ViewCatDetails() *searches for a particular cat and outputs its details*
+UpdateList() *searches for a specific cat to update and then updates the c depending on which attribute is selected to be updated*

Program 10.5

Write a program to draw shapes using console mode digits. Your shapes sh
definitions below.

Class Shape	
Protected char drawletter	*Holds what character will be used in compiling*
Protected string shapeownername	*This is the name of the person wanting this sh*
Protected int shapeheight	*Holds the image h the number of lines to*
Public Shape()	*Constructor method*
Public GetShapeOwnerName()	*Returns who the owner of this shape is*
Public virtual void DrawShape()	*Draws the shape in the console window using*

Class Rectangle Inherits Shape	
Private int width	*Holds the width of the image (the number of ch in the console window)*
Public Rectangle()	*Constructor method*
Public override void DrawShape()	*Draws the shape in the console window using*

Class Triangle Inherits Shape	
Private int startpoint = 1	*Holds the starting width of the triangle (the nu the first line in the console window)*
Public Triangle()	*Constructor method*
Public override void DrawShape()	*Draws the shape in the console window using*

TIP – Code to draw a shape in console mode. This code will output a rectangle

```
int width = 6;
int height = 10;

for (int i = 0; i < height; i++)
{

    for (int j = 0; j < width; j++)
    {
        Console.Write("X");
    }
    Console.WriteLine();
}
Console.WriteLine();
```

Evidence required:

1. **Annotated code listing** showing all three shape classes and a main p select a shape, input its parameters and output the shape. When a sha program should ask the user whether they want to draw another shape
2. **Test screenshots** that show parameters being input for three different being drawn.

C# Quick Syntax Gu

When I want to	The C# statement(s) I need to use	How it
Use variables	**Variables are containers** used to hold values of different data types during the program execution. These **values can change** as instructions are executed. Use INTEGER for whole numbers, DOUBLE for decimal numbers and STRING for text.	At the You ne String int a char
Input values	In console mode, we use the Console.ReadLine() statement to accept an input value into our program. The value is read into the program and assigned to a declared variable.	This is **Write** asked String Consol name
Process values	This will often involve the **assigning of values** from one variable to another and possibly the use of **mathematical or relational** operators.	total
Output values	You can output from a program using the Console.WriteLine() or Console.Write() statements.	Consol
Stop the console window closing	After output you can **stop the console window closing** by using the Console.ReadLine() statement just on its own.	Consol
Generate a random number	This statement allows you to generate a random number.	Rando int v This wi down:

When I want to	The C# statement(s) I need to use	How it
Change the console colour	This statement allows you to change the console window **background** colour.	Consol
	This statement allows you to change the colour of the **text**.	Consol

Make choices	This is known as selection. This structure is used when you want the program to perform an action (or actions) based on the outcome of a test. The test '**tests**' **whether a condition has been met** or not and will perform the following actions if the condition is met (is 'true').	if (co
		{
		co
		}

	Should you wish to run code in two cases, you can use the **else** statement.	if (co
		{
		co
		}
		else
		{
		co
		}

	You can also add more conditions by using the **else if** statement.	if (co
		{
		co
		}
		else i
		{
		co
		}
		else
		{
		co
		}

When I want to	The C# statement(s) I need to use	How it
Compare variables (operators)	To compare variables, we need to use **operators**. The most used operators are used to compare two variables (**relational operators**).	== E != N < L <= L > G >= G
	The next type of operator is used with more than one argument in the same condition (**logical operator**).	&& A Both A A || (Either A ! (Conditi

Make advanced choices	Switch/case statements are used when multiple outcomes are possible.
	
	(**NB** the **break;** statement is needed to avoid code running into the next case.)
		switch(
		{
		cas
		
		cas
		
		cas
		
		}

	You can also use nested statements to have specific outcomes.	if(grad
		{
		if(
		{
		}
		}

	You can also add multiple conditions together. (Please check the operators section for more details).	if(gra
		{
		Con
		}

When I want to	The C# statement(s) I need to use	How it

Repeat the same lines of code (loops)	Loops are used when you want to repeat the same lines of code / operations.
	The first way of doing this is with the for loop.
		To run
		`for (i`
		`{`
		`  Co`
		`}`

	The second method is having a while loop (this may be skipped if initial condition is met).
	Be careful! While loops may run infinite times if not coded correctly.
		`while`
		`{`
		`  Co`
		`  h+`
		`  th`
		`  ev`
		`}`

	The third type of loop is the do/while loop and will run **at least once** before stopping.
	Be careful, do/while loops can also run endlessly!
		`do`
		`{`
		`  Co`
		`  h+`
		`  ch`
		`  ev`
		`}`
		`while`

Use an array	Arrays are used to store a collection of data that have the same data type. The most basic array (and possibly the most common) is the one-dimensional array.	`Stri` The sq creatin determ this me **NB arr** **will ho**
	Arrays can also be declared then initialised on the same line.	`Stri` This m or if all

When I want to		The C# statement(s) I need to use	How it
Add to an array		If you've started with an empty array, you can add values to it manually.	`name`

		The easiest way of doing this is via a [illegible] (You can also set how many items you put to the array manually by accessing the *length* property.)
		
		This will ensure you avoid an array out of bounds error (whilst filling every index.)
			`for (`
			`{`
			`  c`
			`  n`
			`}`

Clear an array		There are times when you need to clear an array of its values. This is done by using *Array.Clear*.	`Array` Inside the sta
Output from an array		When you want to access data in an array, you need to specify which element you want.	`Cons`
Use an advanced array	**(for strings)**	Arrays can be multidimensional! (This is generally used to create tables with columns and rows.)	`Stri` Each d

		Additionally, like 1D arrays, the content can be stated at the point of declaration.	`String`
			`  {`
			
			NB eac contain

When I want to		The C# statement(s) I need to use	How it is

Use an advanced array	**(for different data types)**	**<u>However</u>**, if you want to store different data types inside a multidimensional array (or simply want to have a more detailed structure), you need to create a separate *struct* to inform your program how to store the data outside of the main method.
		
		An array of the struct is then created.
			public
			{
			pu
			pu
			}
			
			This stat
			
			The *stru*
			example
			
			As can b
			differenti
			
			GhostB
			
			Inside th
			array, yo
			array (he
			
			GB[0]
			GB[0]
			GB[1]
			GB[1]
			
			To acces
			array ind

Find the length of a string		Similarly to finding the length of an array, we access the *.Length* property.	Strin Consc **NB the l** **counted**

When I want to	The C# statement(s) I need to use	How it is
Find the position of a value	When looking for a particular value in a string, we use the *.IndexOf* method. This can be used for individual values (chars) or words.	`Consol` Using th position
Insert a value	When you need to insert values into a string at a particular point, we have to use the *.Insert* method with the index position and the data to be inserted.	`text` The outp **NB be c** **method.**
Remove a values	There are times when we need to cut down a string. This is done using *.Remove*.	`Conso` Carrying
Replace a specified value	If a specific value/word needs to be changed, we simply need to use the *.Replace* method.	`Console` This will **NB this** **assigne**
Return a selection of characters	To obtain a substring from a string, we use *.Substring*. This can be used to create new data items.	`String` newText
Check whether a particular value is in a string	An easy way of checking whether a word is in a string, for example, we use *.Contains*. This outputs a Boolean true or false.	`Consol` As newT

Split a string	This can be done to pick out individual data items (such as words) from a string by using a char delimiter.
	
	This is done by using *.Split*.
	
	Using this method allows us to create an array of data items (in this case, the words 'British' and 'Queen').
		`String`
		`foreac`
		`Co`
		
		The deli
		
		This outp
		British
		Queen

When I want to	The C# statement(s) I need to use	How it is
Join strings together	If you want to join different strings together, we use the *.Join* method. The parameters needed are a delimit[illegible] (usually String) and the strings to join (in the example, a comma and the splitText array).	Using th[illegible] `Consol[illegible]` This will [illegible] Here, ea[illegible]

Compare strings	.Compare is used to compare the order of two strings against each other.	`Consol[illegible]`
		`Str[illegible]`
		
		Here, th[illegible]
		
		This is b[illegible] alphabet[illegible]

Read from a file (generic)	To use the contents of a file, we first have to tell the program we're expecting a file by using *System.IO* (for Input/Output).	`usi[illegible]` `usi[illegible]` The *usin*[illegible] input, ou[illegible]

Read from a file (generic)	We then need to tell the program where to find the file.	`strin[illegible]`
		
		By defau[illegible]
		
		If you're [illegible] characte[illegible]
		
		`string[illegible]`
		`"[illegible]`

	Once the program knows where to find the file, we then assign the contents of the file to an object.	`strin[illegible]`
		`F[illegible]`
		
		The *File*[illegible] range of [illegible] of the fil[illegible]

When I want to	The C# statement(s) I need to use	How it is

Write to a file (generic)	The *.WriteAllText()* method opens a file, overwrites it with the given text and then closes the file. In this example, it allows us to take all input from the console and save it into the stated file.
	
	However, using this method can result in writing a lot of information all at once (think of the memory usage!).
		`Console`
		`uname =`
		`File.Wr`
		
		This is g
		creating

	The *.AppendAllText()* method is used to open a file, add the given text to the end of the file and then close the	`Consol`
		`"P`
		`File.A`
		
		This will

Read from a file (advanced)	Using the File methods can use up a lot of system resources. A more efficient method of writing to a file is by using a Stream object.	`Strea`
		`ne`
		
		When re

	Methods such as .ReadLine() can be used to read a line from the file. By putting it in a loop, we can extract multiple lines of information.	`while (`
		`{`
		`Co`
		`}`
		
		The *.Pee*
		data, an

	And then it is [illegible] this ensures that, should the program crash, the file integrity is maintained.	`SR.Cl`
Write to a file (advanced)	[illegible] to a Stream, we use a StreamWriter object.	`Strea` `ne`
	We can then add the latest information to a new line at the end of the file.	`SW.W`
	Once finished, the Stream object is closed.	`SW.Cl`

Answers: Written Tasks

Chapter 1 – page 10

1. Some Internet research should enable students to find other C# data types, which m
   a) byte (for an 8-bit unsigned integer which will store integers from 0 to 255)
   b) sbyte (for a 8-bit signed integer which will store integers from -128 to 127)

   This question is a useful way of getting students to consid the most appropriate da they need to store.

2. Most appropriate data types:
   a) Surname: string
   b) Raw re: float or double
   c) Pe e mark: float or double
   d) Gra char
   e) Passed the exam: bool

Chapter 1 – page 12

1. The two WriteLine statements will produce two different answers. The first WriteLin of 64 and the second an output of 100.
2. The first WriteLine statement has no brackets; therefore, the multiplication (C*A) wi added to whatever the value of B is. The second WriteLine statement has brackets a the addition to be done first (B + C), and the result of this will then be multiplied by A
3. The rules applied are the BODMAS rules.

Chapter 3 – page 20

1. When the break keyword is reached, it causes the switch block to terminate, and pro of code after the switch block finishes.
2. The default keyword provides a default case, so if none of the other cases applies, th is why it is always at the end of the switch statement.

Chapter 5 – page 34

1. 38
2. 7 (0–6)
3. Index 4
4. TestMarks[2] = 61

Chapter 5 – page 36

1. There is a range of other methods – include
   a. GetLength() *returns the nu nents in the array*
   b. Copy() and CopyTo(nents into another array*
   c. Clear(ets nents to zero, false or null*
   d. G) *returns the value at a specified position*
   e. Set e() *sets the value of a specified element*

Chapter 5 – page 39

1. 49
2. 12
3. TestMarks[0,3]
4. TestMarks[2,2] = 94

Answers: Consolidation Tasks

Chapter 1

1.

- *The program uses the `Console.WriteLine` method to output to the conso*
- *The string (text) / values contained in the brackets is what gets displayed i*
- *The string / values in the brackets are passed as arguments to the `WriteL`*
- *The `WriteLine` method will always write to a new line in the console windo method will add to the current line*

2.

- *When the user types a value into the console window, this is read into the `Console.Readline` method*
- *The values may be assigned from the `Readline` method to a variable, or t argument to the `WriteLine` method (so that the value is directly output aga*
- *If a value that is read is passed to a variable of a different data type to st be converted to the correct data type using the parse or `Convert.To` meth*

3. The values are being passed (assigned) to an integer variable; therefore, they converted to an integer data type from a string

4.

- *The data type chosen for the material price was an integer*
- *The value entered for material price is a real number*
- *Therefore the program crashes because it is the wrong data type*
- *To avoid this, the data type of the material price needs changing to a doub*

Chapter 2

Questions:

1. Because this is a value that will not change during the execution of the progra

2. The value of a constant does not change during the execution of a program, w the value of a variable can change many times during the execution of the pro

3. It makes the program easier to follow and, therefore, update when required.
This is particularly true when a program is being updated by a different progra

4. Declaration, initialisation and assignment, casting

5. Not all programming software makes it mandatory to declare variables.
In C# the declaration statement is used to reserve a place in memory for that

6. Scope refers to the part of the program where a variable can be seen and acc
A variable with local scope can only be seen in the block of code (procedure, declared in which it is passed as an argument or as a parameter to a called fu its nor scope
A variable with global scope can be seen and accessed by the whole program

7.
i. String enteredpassword
ii. String savedpassword
iii. Int attempts
iv. Bool match

Coding Consolidation Tas
Generic Marking Guidan

Student responses to the programs set in the consolidation tasks can be marked us
as the maximum available mark. This is to reflect the fact that there is often a num
solutions to a given task. One example solution has been provided for each consoli

18–20	• A **working solution** to the problem has been developed that m requirements of the task or problem. • **Testing** shows that the correct respon e achieved using th (where appropriate) abnormal and b ndary inputs. • The **code** used to pr olution to the problem is **efficient** *the most a pri statements and structures for the task: us mul s; ing the most appropriate loop for the task; for th co into functions and procedures and using appropriate para local rather than global variables, etc.)*
13–17	• A **working solution** to the problem has been developed that m requirements of the task or problem. • **Testing** shows that some required responses are achieved usin • The **code** used to produce a solution to the problem is mostly e
9–12	• A **solution** to the problem has been developed that meets **mos** requirements of the task or problem. • **Some testing has been carried out, although this may not pro** • The **code** used to produce a solution to the problem has **some** students may not always have selected the most appropriate st
1–8	• A **partial solution** to the problem has been developed that mee requirements of the task or problem. • **There may be little or no testing.** • The **code** used to produce a solution to the problem shows an follow a line of reasoning and select **some** appropriate stateme
0	• No markworthy material.