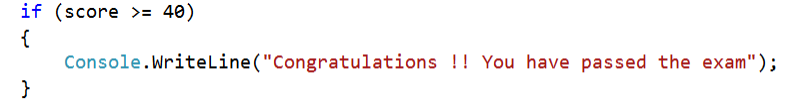
# Chapter 3 – Using Selection

## What is selection?

Selection refers to the fact that one or more lines of code may or may not be executed depending on the outcome of a condition (test) being true or false. Selection statements use conditional logic.

****

This conditional statement checks whether the value of score is more than or equal to 40. This will be either true or false. If true, then the message in the curly brackets will be output. If false, then nothing 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 linear (sequenced) fashion from beginning to end. Real-life programs obviously need more versatility and flexibility than that. They need the ability to branch off in different directions, depending on values or user input. Selection gives us this ability to build decision-making into our programmed solutions to problems.

A simple example might be the need in a password checker program to output one message if an 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 conditional statements.

|  |  |  |
| --- | --- | --- |
| **Relational operator** | **Explanation** | **Syntax example** |
| **==** | Equal to | if (score == 40) |
| **>** | More than | if (score > 40) |
| **>=** | More than or equal to | if (score >= 40) |
| **<** | Less than | if (score < 40) |
| **<=** | Less than or equal to | if (score <= 40) |
| **!=** | Not equal to | if (score != 40) |

## Types of selection construct and C# syntax

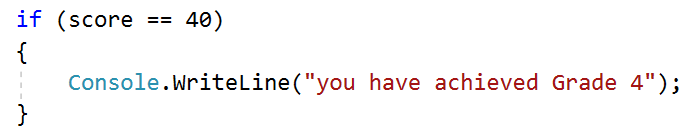
C# has a number of different selection constructs.

**IF…**

This statement is used when you only want code to execute if the outcome of the condition is true, and no code is to be executed when the outcome of the condition is false. In this case, the program execution will continue with any code statements outside the IF block.

The example below shows how an IF statement is written.

**IF** keyword is followed by the condition to be used in curved brackets.



The condition is followed by a set of curly brackets which contains the code to be executed if the outcome of the condition is true. In this case, the message will be output only if the value of score is equal to 40.

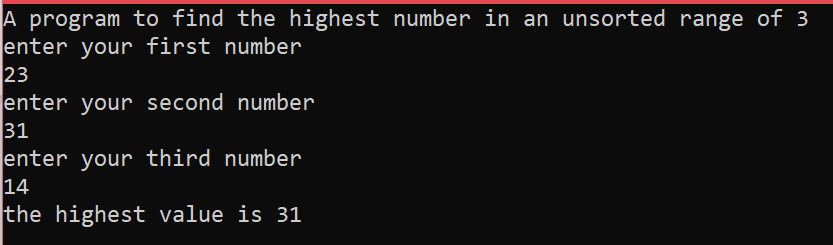


**{ }**

Coding Task: *Program 3.1*

Write a program that will find the highest number out of an unsorted range of three integers.

The screenshot below shows the expected output.



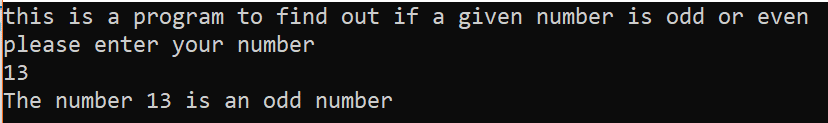


**{ }**

Coding Task: *Program 3.2*

Write a program that will output whether a given integer number is odd or even.

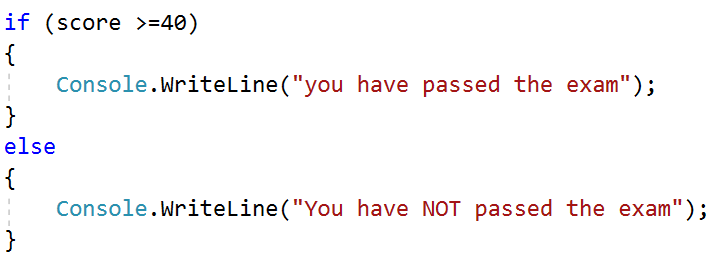
The screenshot below shows the expected output.



**IF ELSE…**

This type of selection construct has code to be executed if the outcome of the test is true and an alternative block of code if the outcome of the test is false.

Keyword **if** followed by the condition to be used in curved brackets.



The condition is first followed by a set of curly brackets which contains the code to be executed if the outcome of the condition is true.

The **else** keyword is used to indicate that the block of code following is to be executed if the outcome of the condition is false.

**Logical operators for joining conditions**



**{ }**

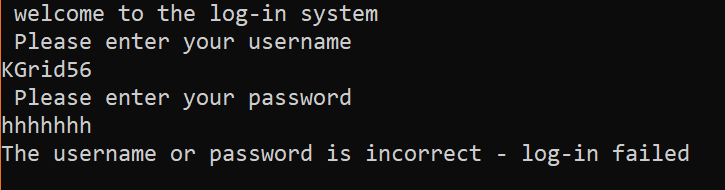
|  |  |  |
| --- | --- | --- |
| **Logical operator** | **Explanation** | **C# syntax** |
| && | AND | (numA ==1 && numb == 2) |
| || | OR | (numA ==1 || numA == 2) |
| ! | NOT | (numA != 2) |

Coding Task: *Program 3.3*

Write a program that checks whether a username and password used to log into a computer system   
are both correct. The correct username is ‘KGrid34’ and the password is ‘KG7&56@’.

If the user attempt to log in is successful, then output a message ‘you are logged in’. Otherwise, a message ‘The username or password is incorrect – log-in failed’ should be output.

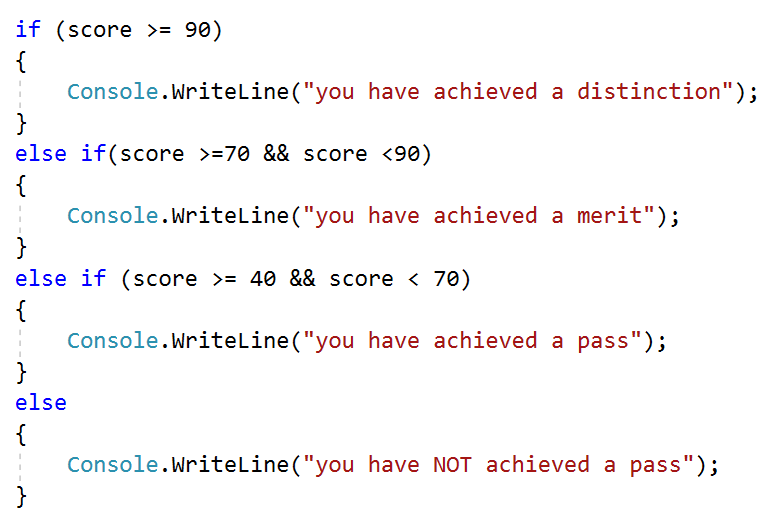
The screenshot below shows the expected output for an incorrect password.



**IF ELSEIF ELSE…**

The ELSEIF part of the statement is used to add further conditions and alternative outputs or instructions.

The **else if** keywords are followed by a new condition to be tested and a block of code to be executed only if the outcome of this new condition is true.



The else statement at the end of an **if** and one or more **else ifs** is used to provide code to be executed if none of the conditions evaluates to true.

Each **else if** will only be executed if the preceding **if** or **else if** condition evaluates to false.



**{ }**

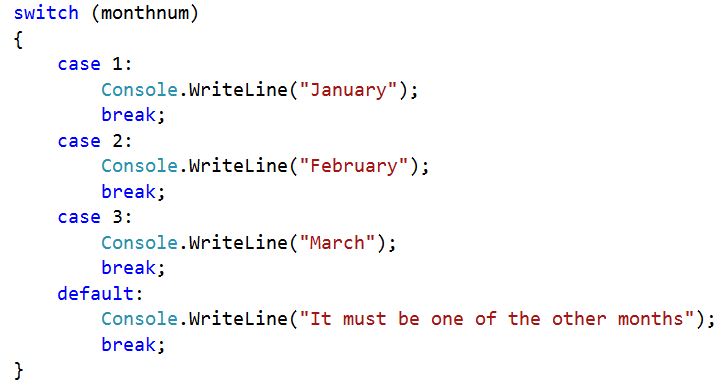
Coding Task: *Program 3.4*

Write a program that takes in a student’s marks for an exam and outputs the grade attached to that mark. An A grade needs 80 or more marks, a B grade needs 70 or more marks, a C grade needs 55 or more marks, a D grade needs 45 or more marks, and an E grade needs 30 or more marks.

**SWITCH CASE**

SWITCH is used when you have either multiple cases with a particular output or a set of statements to be executed depending on the case.

**monthnum** is the value to be passed into the switch statement – it is this value that is checked against each case





In this case, if **monthnum** = 2 then the program will output ‘February’ to the console window.

Written Task

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



**{ }**

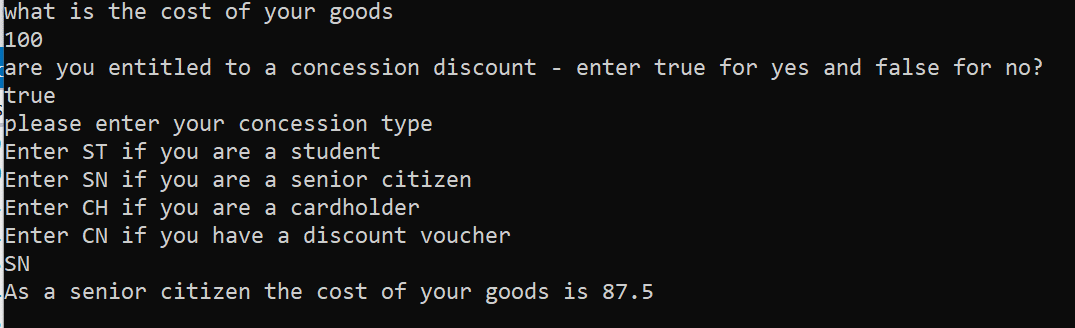
Coding Task: *Program 3.5*

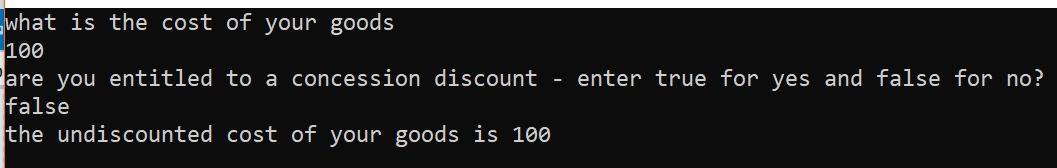
A department store offers various discounts, based on the following discount codes:

* (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 final cost after the appropriate discount has been applied, based on the value of an input discount code. No discount should be applied if the customer is not entitled to a discount. You must use a switch statement.

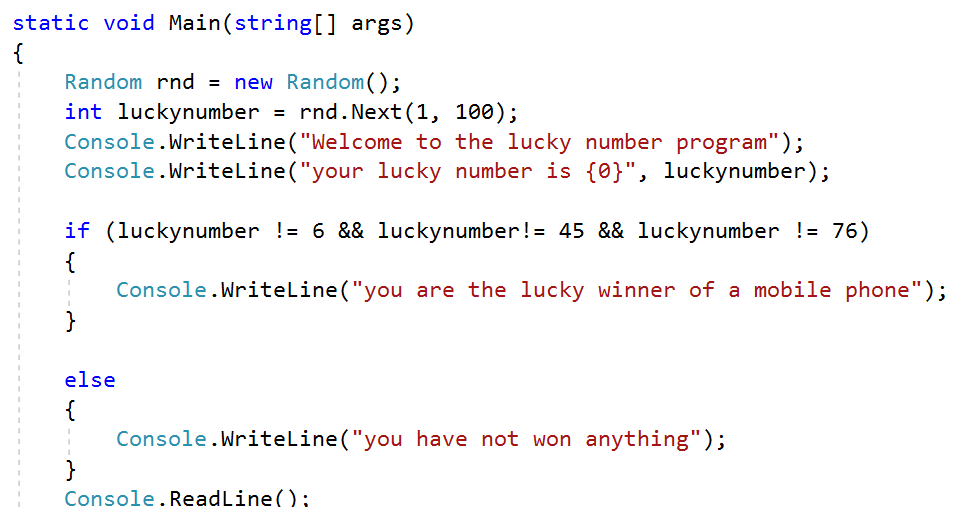
The console window examples below show the expected output.





## Multi-condition statements

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



## Chapter 3 – Consolidation Tasks

Program 3.6: *Insurance program*

Insurance Plus is an insurance broker based in Staffordshire. It offers discounts to local customers. It has had the program **“Program 3.6.txt”** written for it and now wants you to add some extra features. 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 age and postcode. Customers under 25 are charged double the basic premium. Customers who live in an ST postcode are entitled to a 10% discount on their premium. Customers over 25 are charged the basic premium. Customers who live outside the ST postcode area do not receive a discount.

**Evidence required:**

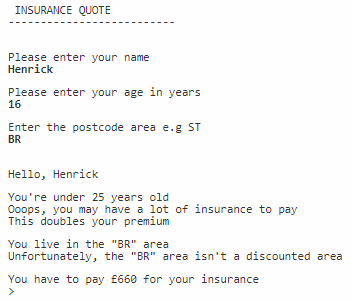
1. Annotated code listing for the amended program.
2. Testing screenshots of the following tests:
3. A customer under 25 with an ST postcode
4. A customer over 25 with an ST postcode
5. A customer under 25 with a BR postcode
6. A customer over 25 with a PR postcode

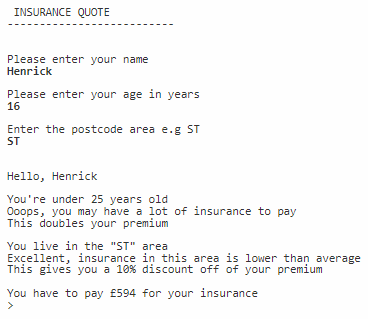
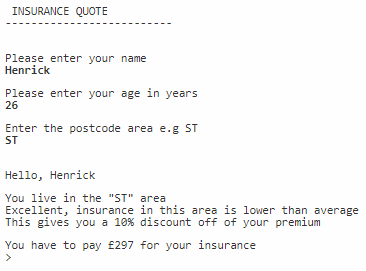
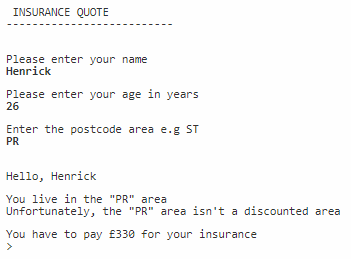
## WRITTEN TASK

(Under program 3.4)

1. The keyword **break;** stops the program for that “case” so that you can go to the next case
2. The keyword **default** tells the code what to do if the variable doesn’t match any of the other cases

## Program 3.6

(Insurance program)

1. **I will post the program.cs file for program 3.6 with this word document**