

Practical Task 1.1

1. Create a new Console Application project and write a program to print a number to console as a word. This program is to replace for you a traditional 'Hello World' example. It is required to accept an integer value as input (i.e. numbers 1 to 9), and then print the corresponding word on screen (i.e. "ONE", "TWO", ... , "NINE"). If anything other than 1 to 9 is entered, then an error message is displayed. This simple program is used to make you thinking about validation; that is, has the user input only integer values between 1 and 9? If not, then we need to output an error message.

Give a new name to the main class by renaming the default Program class name (and the associated file name) to IfStatement in the Solution Explorer of your IDE (Microsoft Visual Studio or Studio Code). Type the following code into the main IfStatement class and complete the other **else if** statements.

```
static void Main(string[] args)
{
    Console.WriteLine("Enter the number (as an integer): ");
    int number=Convert.ToInt32(Console.ReadLine());
    if (number == 1)
    {
        Console.WriteLine("ONE");
    }
    else if (number == 2)
    {
    }
}
```

Complete the missing parts of the program for the remaining numbers 2 to 9. Compile and run the program, then examine the produced output. You should aim to test your program with the following:

All nine integers between 1 and 9.

A few other integers outside this range, i.e. 10 and 100.

What happens if you enter -5?

What happens if you enter a letter, e.g. 'a'? How can you add an error message to catch this? (HINT: We normally use a **try-catch block** to catch the error. Do not worry if you are unable to complete this now, we will look at it in more detail later on in the unit.)

2. Now, create a new project and name its main class (and file) as SwitchStatement. Implement another version of the above program by writing the following code.

```

Console.WriteLine("Enter a number (as an integer): ");
int number = Convert.ToInt32(Console.ReadLine());

switch (number)
{
    case 1: Console.WriteLine("One"); break;
    case 2: Console.WriteLine("Two"); break;

    default: Console.WriteLine("Error: you must enter an integer between 1 and 9"); break;
}

Console.ReadLine();

```

Complete the rest of the case statements up to 9 (between case 2 and default). Run the program and check whether its output is as expected. Test it with the following:

All nine integers between 1 and 9.

A few other integers outside this range, i.e. 10 and 100.

What happens if you enter -5?

What happens if you enter a letter, e.g. 'a'? How can you add an error message to catch this?

3. Find all existing issues with the code snippets presented below. It is highly important to try working this out on paper by tracing through the statements and then input the code into your IDE. Run it to see if you are correct, and then see if you can fix the code.

```

int number = 50;
if (number == 50) ; {
    Console.WriteLine("Number is 50");
}

```

```

int number = 60;
if (number >= 50 and number <= 100) {
    Console.WriteLine("Number is more than or equal to 50 and less than or equal to 100");
}

```

```

public class Score {
    public static void main(String[] args) {
        double score = 40;
        if score > 40{
            Console.WriteLine("You passed the exam!");
        } else score < 40{
            Console.WriteLine("You failed the exam!");
        }
    }
}

```

```

Switch (n) {
    case 1: Console.WriteLine("The number is 1");
    case 2: Console.WriteLine ("The number is 2"); break;
    default: Console.WriteLine ("The number is not 1 or 2"); break;
}

```

```

switch (n) {
    Case 1: Console.WriteLine ("A"); break;
    case2: Console.WriteLine ("B"); break;
    Default: Console.WriteLine ("C"); break;
}

```

4. What is the output of the following code fragments? Remember to try working this out on paper by tracing through the statements and then input the code into. Run it to see if you are correct, and then see if you can fix the code.

```
int height = 13;
if ( height <= 12 )
    Console.WriteLine("Low bridge: ");
    Console.WriteLine ("proceed with caution.");
```

```
int sum = 21;
if ( sum != 20 )
    Console.WriteLine ("You win ");
else
    Console.WriteLine ("You lose ");
    Console.WriteLine ("the prize.");
```

```
int sum = 7;
if ( sum > 20 ) {
    Console.WriteLine ("You win ");
} else {
    Console.WriteLine ("You lose ");
}
    Console.WriteLine ("the prize.");
```

5. A microwave oven manufacturer recommends that when heating two items you should add 50% to the heating time, and when heating three items you should double the heating time. Heating more than three items at once is not recommended.

Create a new project and name its main class (and file) as *Microwave*. Write a program that asks the user for the number of items and the single-item heating time. Perform the calculation, and print out the recommended heating time.

6. This exercise requires you to cast the result of an operation performed on two *integers* to a *double*. You should have already used casting to convert a *String* to an *int*. via

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

as a combination of the two commands. The compiler will not let you assign a value to a variable if it is the wrong type – even if that variable can hold the value. When you use casting, you are essentially making a promise to the compiler that you know the types are different and that in this particular instance it is alright for C# to push data into a new variable.

Create a new project and name its main class (and file) as *DoCasting*. Write a program by following the steps below:

Declare and initialise variables for two integer values **sum** and **count**.

Initialise these variables to 17 and 5 respectively.

Declare an *integer* variable called **intAverage**.

Calculate the integer average, **intAverage**, using **sum/count**.

Print out the integer average, **intAverage** – is this correct?

Declare a *double* variable called **doubleAverage**.

Now try calculating the double average, **doubleAverage**, using **sum/count** – what is the result? Is this correct?

Cast the **sum** variable to a double using **(double)sum** and divide this by **count**, and print the result – is this correct?

Remember that the process for casting is to place the data type in parentheses before the operation that is being performed, which in this case is **sum/count**.

7. Design an algorithm, first on paper, using pseudocode which prints the sum of 1, 2, 3, ..., to an upper bound determined by user input (e.g. 100). Then create a new C# Console Application project and write a program to achieve this. Once coded, it should also compute and display the average of the numbers. The output should look like:

- The sum is 5050
- The average is 50.5

Give a new name to the main class of the project by changing it from the default Program to Repetition in the Solution Explorer of your IDE (Microsoft Visual Studio or Studio Code). Type the following code into the Main method of the Repetition class.

```
static void Main(string[] args)
{
    int sum = 0;
    double average;
    int upperbound = 100;

    for (int number = 1; number <= upperbound; number++)
    {
        sum += number;
    }

    //Compute the average as a double - remember that int divided by int produces an int
    //Print the sum and the average and check that your results are correct
}
```

Complete the missing parts, compile and run the program. Check whether the produced output is as expected.

8. Add a new block of code into the Main method. Modify the code written in the first task to use a **while loop** instead of the **for loop**, which you need to replace with the following code.

```
int number = 1;
while (number <= upperbound)
{
    sum += number;
    number++;
}
```

What is the difference between **for** and **while loops**? If you want to see what is happening in the loops then trace **sum** and **number**. You can do this by adding the following line inside each of the loops after the sum calculation:

```
Console.WriteLine ("Current number: " + number + " the sum is " + sum);
```

This will allow you to see what is happening during each iteration of the loop.

9. Add one more block of code into the **Main** method. Modify the code to use a **do ... while loop** instead of a **while loop**. Replace the **while loop** with the following code.

```
int number = 1;
do
{
    sum += number;
    number++;
}while(number<=100);
```

Check that the output from all three loops is the same. What is the difference between **for**, **while** and **do ... while loops**?

10. Find existing issues for all the code snippets presented below. It is highly important to try working this out on paper by tracing through the statements and then input the code into your IDE. Run it to see if you are correct, and then see if you can fix the code. You may wish to add additional `Console.WriteLine` statements to check that variables contain the values you expect. Note that there may be more than one mistake.

```
int c = 0, product = 0; while (c
<= 5)
    product = product * 5; c
    = c + 1;
```

```
int a = 31, b = 0, sum = 0;
while (a != b) {
    sum = sum + a; b
    = b + 2;
}
```

```
int x = 1; int
total;
while (x <= 10) { total =
    total + x; x = x + 1;
}
```

```
while (y < 10) { int
    y = 0;
    Console.WriteLine("y" + y); y =
    y + 1;
}
```

```
int z = 0; while
(z > 0) {
    z = z - 1; Console.WriteLine("z: "
    +z);
}
```

```
for(int count = 1, count < 100, count++) { Console.WriteLine("Hello");
}
```

```
for(int i =1; i>10; i++) { if
    (i>2) {
        Console.WriteLine ("Flower");
    }
}
```

11. Rewrite both of the following fragments of code using *for loops* instead of *while loops*.

```
int sum = 0 ;
int j = -5 ;
while ( sum <= 350 ) {
    sum += j ;
    j += 5 ;
}
```

```
int x = 0;
while ( x < 500 ) {
    Console.WriteLine( x );
    x = x + 5;
}
```

12. Write a program that will continue to prompt the user to guess a number until the correct number has been entered. You should also consider user validation; that is, has the user entered a number less than 1 or greater than 10? Have they entered a character? Start by creating a new project and name its main class (and file) as `GuessingNumber`. Then proceed with the following steps:

Create an integer variable called **number** and set it equal to 5.

Prompt the user to guess a number between 1 and 10.

If the user enters number 5, print "You have guessed the number! Well done!"

If they do not enter 5, continue to prompt the user to enter a number until they enter

- 13.. Compile, run and check the program for potential errors.

Now, alter the program so that it takes a number entered by a user (user 1). This number is set as the number to be guessed by user 2. Then ask (user 2) to input their own number. Check to see if the number user 2 entered is equal to the number user 1 entered. Continue this process until the user 2 guesses the correct number. Remember that for this program you should validate the user's input. Finally, run and check the program again.

14. Write a program which prints all the numbers between 1 and n, where n is provided by the user, that are evenly divisible by four but not by five. You should use a **for loop** for this question.

Create a new project and name its main class (and file) as `DivisibleFour`. Compile and run the program. Check whether the produced output is as expected.

