C# Exercises – Part 1

# Summary

These exercises are meant to reinforce knowledge from the “Software Engineering Fundamentals in C# - Part 1” PowerPoint.

# Hello world

1. Using Visual Studio, create a new console application
   1. Name both the solution and the project “HelloWorld”
2. In the Main method, print “Hello World” to the console

# Comments

1. Create a single line comment
2. Create a multi-line comment block
3. Create an XML documentation comment

# Variables

1. Declare a bool, an int, a double, a char, and a string
2. Initialize all variables with values
3. Print the values to the console

# Min / Max values

1. Declare an int and initialize it to the max value
2. Increment the int by one
3. Observe what happens

# Implicit types

1. Declare a bool, an int, a double, a char, and a string all using “var”

# Operators

## Assignment and Increment/decrement

1. Declare integers x and y and initialize them to any values
2. Declare z1 = x++;
3. Declare z2 = ++y;
4. Print the values of x, y, z1, and z2

## Math

1. Declare integers x, y, and z, and initialize them to any values
2. Declare integers a, b, and c, and initialize them to any combination of math operations using x, y, and z
3. Write an expression that returns true if x is even, and false if x is odd. Store the result of the expression in a bool.
4. Write an expression that returns false if y is divisible by 3. Store the result of the expression in a bool
5. Write an expression that rounds a double to 3 decimal places and store the result in a double.

# Type conversions

1. Declare integers x, y, and z and initialize them to any values
2. Declare doubles a, b, and c and initialize them to any values
3. Store the integers in the doubles
4. Store the doubles in the integers

# Strings

## Escapes and literals

1. Create a string with escape sequences in it and print it out
2. Create a literal string and print it out

## Concatenation

1. Concatenate two strings together and print them out

## Length

1. Print a string along with its length

## Indexing

1. Create a string with length > 3 with an odd number of characters.
2. Print the first, middle, and last character, concatenated together.
3. Make sure your code still works when you change the string size to a different odd number.

## Changing case

1. Print a string, the same string in all uppercase, and the same string in all lowercase

## Getting parts of strings

1. Create a string with length > 3.
2. Print the string with both the first and last characters removed.
3. Make sure your code still works if you change the string size.
4. Repeat the above exercise using a different string method.

## Searching

1. Declare strings x and char y and initialize them to whatever you like.
2. Write an expression that returns the index of the first occurrence of char y in string x
3. Write an expression that returns a bool that represents if y is anywhere in string x
4. Create a literal string that holds a fake path to a file (@”c:\foo\bar.txt”, for example). Write code to get just the file name (no path), without the extension. Write code to get just the drive letter, in uppercase.

## Splitting

1. Declare string x and initialize it to whatever you like. Make sure the string has spaces in it.
2. Split the string on space and observe the results

## Formatting

1. Create a const string x that contains formatting tokens
2. Create another string y that uses string.Format to insert values into string x
3. Print string y
4. Create a double d and initialize it to a number with > 2 decimal places.
5. Print the double, but only include 2 decimals when printing

## Printing and parsing

1. Create bool a, int b, double c, and char d, and initialize them to whatever you want (other than default values).
2. Convert these types to strings w, x, y, and z.
3. Create bool e, int f, double g, and char h. Use the appropriate Parse() method to initialize these variables from the strings w, x, y, and z.

# Enums

1. Create an enum inside the HelloWorld namespace, but outside of the Program class
2. Call the enum whatever you like, populate it with whatever names and values you like
3. Declare a variable and assign it to one of the enum values.
4. Using the variable, print both the enum name AND the enum value

# Structs

1. Create a struct inside the HelloWorld namespace, but outside of the Program class
2. Call the struct whatever you like and add some members to it
3. Declare a variable x and instantiate the struct
4. Declare a variable y and instantiate the struct
5. Set every member in x to the values of every member in y
6. Print the members of x

# Console I/O

1. Create a program that prompts the user for their first name, then middle name, then last name, all on different lines.
2. Print the first, middle, and last name all on the same line
3. Print “Press any key to exit…” and wait for the user to press a key to exit.

# Command-line arguments

1. Create an enum called Direction with the values: Up, Down, Left, and Right
2. Create a program that expects three arguments – a Direction, a bool, and an int.
   1. If the program is called with no arguments OR called with one argument of “/?”, print a help message explaining what arguments are expected with an example of a valid command line.
   2. If either of the first two arguments are missing, print an error message. The third argument is optional
3. Store the arguments in variables of the appropriate type
   1. If any value can’t be stored correctly, print an error message
4. Print the variables stored in step 3. Make sure to include the int if it was specified.

## Visual Studio arguments

1. Test the code above by passing in arguments through visual studio

## Command line arguments

1. Test the code above by passing in arguments directly on the command line