Pre – C# Fundamentals

Course reference

This booklet contains a summary of what you will learn during this preparation course for C# Fundamentals.

Why a programming language?

- People express themselves using a language that has many words, but in the world of a computer, the language is much simpler; it consist of two words only, 1 – stands for "on", and 0 – stands for "off".
- This means that if we would try to talk to a computer in its own language, it would be like trying to use Morse code when talking to your friends. It works, but why?
- Therefore, we have a programming language that basically works as a translator between you and the computer.
- In order to avoid learning computer's native language (known as *machine language*), which might be difficult, we can use a programming language to instruct the computer in a way that is easier to learn and understand.
- To our help we have a specialized program – a *complier*, which converts the code you have written to *machine code*. You do not need to understand machine code, only the programming language that you are using.

Similarities with our language

- We can compare a book to a programming language. It consists of *chapters*, *paragraphs*, *sentences* and *words*.
- In C# we will have a slightly different structure. A *module* is like a chapter, a *procedure* is like paragraph, and a *line* is like a sentence.

Grammar

- In a spoken language, we often have some kind of grammar, e.g., nouns and verbs.
- In a programming language such as C# we have *statements*, *declarations*, *methods operators*, and *keywords*. These are called *programming elements*.
- A language that we speak has a *syntax*, so do C#.

The maximum speed of my car is 55.	Car.Speed.Maximum = 55;

C# Reference

Useful commands

`Console.WriteLine(”text to write”);`	Writes ”text to write” on the screen.
`Console.ReadLine();`	Pauses the program until the user has entered any text and/or pressed enter.
`Console.Clear();`	Clears all text from the screen.
`string name = ”Bob”;`	Declares a variable, and assigns it the value ”Bob”.

Relational operators

`==`	Checks whether two objects are equal.
`!=`	Checks whether two objects are not equal.
`<`	Less than.
`>`	Bigger than.
`=<`	Less than or equal.
`>=`	Bigger than.

Logical operators

`&&`	And
`||`	Or
`^`	Xor
`!`	Not

Data types (also Primitive types)

`Int32 (or int)`	32 bit integer
`Int64 (or long)`	64 bit integer
`Boolean (or bool)`	True or false
`Float (or float)`	Single precision floating point
`Double (or double)`	Double precision floating point
`Decimal (or decimal)`	Fixed precision floating point (financial)
`DateTime`	An instant in time (to 100 ns)
`String (or string)`	Text (as Unicode[1] characters)

Integers – when you want to work with whole numbers.
Float & Double – when you want to perform scientific calculation (e.g. 5.81×10^{8}).
Decimal – when working with financial calculations (i.e. money).

[1] Unicode allows us to see a text of a specific *writing system* in the same way, on all machines in the world.

Variables

```
int age;                    declares a variable. (not recommended)

int height = 0;             declares a variable, and assigns it the value 0.
age = 56;                   sets the age variable a new value, 56.

Console.WriteLine(age);     writes the value of age on the screen.
```

Note: When trying to set and/or retrieve a value from a variable, please be sure that it is declared! When declaring a variable, please also assign it a value, e.g. *int height = 0;*

Arrays

```
int[] age;                      declares an empty array. (not recommended)

int[] roads = new int[10];      declares an array with 10 empty elements.

int[] height = {100, 189, 123, 153, 190};  declares an array with 5 elements,
```

where each element has a value, e.g. 100, 189, 123, 153 and 190.

```
int[] height2 = new int[] {100, 189, 123, 153, 190};  exactly the same as
```

height, but instead, we explicitly tell the compiler that it's a new instance of that object.

Note: As with variables, when trying to retrieve a value from an unsigned variable, the C# complier will through an exception (i.e. error). The best, and the quickest way of declaring a new integer array is as height is declared, e.g. *int[] height = {100, 189, 123, 153, 190};*

Functions

```
int MyFunction(int x, int y)
{
      return (x + y) / 2;
}
```

Note: A function, as in mathematics, may take several input parameters (values), but, it might only return one value. Please note, a functions may take different kinds of *data types* as well, i.e. bool, float, double, decimal, etc, and even arrays.

Methods

```
void SayHello(string name)
{
      Console.WriteLine("Hello" + name);
}
```

When you call this method, it will output *Hello youname*. Note: the main difference between a function and a method is only the return value. *A method does not return a value*. Please compare *Ref-21* to *Ref-22*. In order to execute a function, we need to specify where to put the result, e.g. in *Ref-21*, the *result* variable is declared, to store the output from *Add*.

Conditional Statements

```
if (condition)
{
	//...
}
else if (condition2)
{
	//...
}
else
{
	//...
}
```

Note: If-statement will execute the *condition1* first, and if it is true, it will execute code inside the brackets. If not, it will continue to check whether *condition2* is true, and perform the same procedure. If both *condition1* and *condition2* are false, it will execute *else*, if there is one.

For loops

```
for (int i = 0; i < index; i++)
{
	//..
}
```

Note: For loop will first declare a variable, *i*, and assign it the value 0. It will then check, whether the *condition*, in this case *i* < *index* is true, and if so, it will execute the code inside it. When that is done, the *i* will be increased by 1, in *i++*.

While

```
while (condition)
{
	//...
}
```

Do While

```
do
{
	//...
} while (condition);
```

Note: the only difference between a *While loop*, and a *Do While loop* is that the while loop will first check whether the condition is true, and later execute the code inside. *Do While loop* will directly execute the code inside it, and only then see if the condition is true. Please take a look at *Ref-14*.

Exercises

Preparation 1:
Please do following:

1. Create a new project (File > New Project).
2. Try to create a program that outputs *Hello World*.
3. Make the program to read the input text from the screen, and output it with any prefix, e.g. "What is your name?", "Hello your_name".

Note: you can always see the right solution at *Ref-1or Ref-3*.

Preparation 2:
Task: *Create an application that will check the age variable*[2]*, and do following:*

- *If the age is less than 10 but bigger than 1, it should write "You are too young!"*
- *If the age is less than 20, it should write "You are an adult!"*
- *If the age is less than 1, it should write "You are not old!"*

When doing this application, please do not forget that an if-statement is executed from top to the bottom, i.e., it will first check whether the condition at the top is true, and if not, it will continue check the second, and so on, until it finds a condition that is true. If it does not find any other condition that is true, it will, execute the *else*, if it is given.

Note: if it is too difficult, please get inspired by *Ref-5*.

Preparation 3:
Please answer these questions:

1. What is a variable?
2. What can be stored in a variable?
3. How do you declare a variable of data the type *integer*, with the value 78, and a name *aNumber*?
4. Can an *integer* contain *infinite* amount of numbers?
5. What might happen if you try to use the variable before it has been assigned a value?
6. If you are developing a system that will calculate the salary, after the taxes, of each employee in the company, what *data type* will you preferably use?
7. If you are creating a *while-loop* that will iterate 25234540 times, what *data type* are you to use?

Please also try to predict the output you will get when executing *Ref-18*. Why do you think it behaves as it does?

Preparation 4:
Task1: *Create a function that will calculate the mean value of an integer array*.
Hint: *You will need to create some kind of loop. (Ref-21)*

Task2: *Create a method that will display "Number - 1", "Number - 2"..."Number-100".*

[2] A variable can be created by typing: `int age = 0;`

Code Reference

Ref-1

```
        static void Main(string[] args)
        {
            string helloMessage = "Hello World";
            Console.WriteLine(helloMessage);
        }
```

Ref-2

```
        static void Main(string[] args)
        {
            Console.WriteLine("What is your name?");
            string name = Console.ReadLine();

            Console.WriteLine("Hello " + name);

            Console.ReadLine();
        }
```

Ref-3

```
        static void Main(string[] args)
        {
            Console.WriteLine("hi"); // writes on the screen

            string name = "world"; // saves text into memory

            string input = Console.ReadLine(); // saves text from the screen

            Console.ReadLine(); // pauses the program
        }
```

Ref-4

```
        static void Main(string[] args)
        {
            int age = 19;

            if (age < 18)
            {
                Console.WriteLine("You are too young to drink alcohol!");
            }
            else
            {
                Console.WriteLine("Still, alcohol is not healthy, drink juice
instead!");
            }

            Console.ReadLine();

        }
```

Ref-5

```
static void Main(string[] args)
{
    int age = 19;

    if (age < 3)
    {
        Console.WriteLine("Serve milk");
    }
    else if (age < 18)
    {
        Console.WriteLine("Serve juice!");
    }
    else
    {
        Console.WriteLine("Serve wine");
    }

    Console.ReadLine();
}
```

Ref-6

```
static void Main(string[] args)
{
    Console.WriteLine("What is your age?");

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

    if (age < 3)
    {
        Console.WriteLine("Serve milk");
    }
    else if (age < 18)
    {
        Console.WriteLine("Serve juice!");
    }
    else
    {
        Console.WriteLine("Serve wine");
    }

    Console.ReadLine();
}
```

Ref-7

```
static void Main(string[] args)
{
    while (true)
    {
        Console.WriteLine("What is your age?");

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

        if (age < 3)
        {
            Console.WriteLine("Serve milk");
        }
        else if (age < 18)
        {
            Console.WriteLine("Serve juice!");
        }
        else
        {
            Console.WriteLine("Serve wine");
        }

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

Ref-8

```
static void Main(string[] args)
{
    int temperature = -1;

    if (temperature < 10)
    {
        if (temperature < 0)
        {
            Console.WriteLine("Take a winter jacket!");
        }
        else
        {
            Console.WriteLine("It's cold, take a warm jacket!");
        }
    }
    else
    {
        Console.WriteLine("No jacket required!");
    }

    Console.ReadLine();
}
```

Ref-9

```
static void Main(string[] args)
{
    int temperature = -1;

    if (temperature < 10)
    {
        if (temperature < 0)
        {
            Console.WriteLine("Take a winter jacket!");
        }
        else
        {
            Console.WriteLine("It's cold, take a warm jacket!");
        }
    }
    else
    {
        Console.WriteLine("No jacket required!");
    }

    Console.ReadLine();
}
```

Ref-10

```
static void Main(string[] args)
{
    //IN WORDS:
    //  if the temperature is bigger than 0 AND if the temperature
    //  is less than 10, then, write "It's cold, take a warm jacket!"
    int temperature = -1;

    if (0 < temperature && temperature < 10)
    {
        Console.WriteLine("It's cold, take a warm jacket!");
    }
    else
    {
        Console.WriteLine("No jacket required!");
    }

    Console.ReadLine();
}
```

Ref-11

```
static void Main(string[] args)
{
    for (int i = 0; i < 100; i++)
    {
        Console.WriteLine(i);
    }

    Console.ReadLine();
}
```

Ref-12

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

    while (i < 100)
    {
        Console.WriteLine(i);
        i++;
    }

    Console.ReadLine();
}
```

Ref-13

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

    do
    {
        Console.WriteLine(i);
        i++;
    }while(i < 100);

    Console.ReadLine();
}
```

Ref-14

```
static void Main(string[] args)
{
    while (2 == 3)
    {
        Console.WriteLine("You will not see this text!");
    }

    do
    {
        Console.WriteLine("You will see this text!");
    } while (2 == 3);

    Console.ReadLine();
}
```

Ref-15

```
static void Main(string[] args)
{
    double aDouble = 6.9M;
    string aName = "Paul";
    bool YesOrNo = false;
}
```

Ref-16

```
static void Main(string[] args)
{
    // this will output the number 12 on the screen.
    int nameOfVariable = 12;
    Console.WriteLine(nameOfVariable);
    Console.ReadLine();
}
```

Ref-17

```
        static void Main(string[] args)
        {
            //NOTE: Arrays are zero based, e.g. the first item
            //has the index 0.

            int[] heightOfStudents = new int[] { 171, 158, 184, 134, 185};

            for (int i = 0; i < heightOfStudents.Length; i++)
            {
                Console.WriteLine(heightOfStudents[i]);
            }

            Console.ReadLine();
        }
```

Ref-18

```
        static void Main(string[] args)
        {
            // predict the output. what element will be changed?

            int[] heightOfStudents = new int[] { 171, 158, 184, 134, 185};

            heightOfStudents[2] = 120;

            for (int i = 0; i < heightOfStudents.Length; i++)
            {
                Console.WriteLine(heightOfStudents[i]);
            }

            Console.ReadLine();
        }
```

Ref-19

```
    class Program
    {
        static void Main(string[] args)
        {
            int result = AddNumbers(5, 9);
        }

        static int AddNumbers(int number1, int number2)
        {
            return number1 + number2;
        }
    }
```

Ref-20

```
class Program
{
    static void Main(string[] args)
    {
        int result = AddNumbers(5, 9);
    }

    static int AddNumbers(int number1, int number2)
    {
        return number1 + number2;
    }
}
```

Ref-21

```
class Program
{

    static void Main(string[] args)
    {
        int result = Add(new int[] { 2, 3 });
    }

    static int Add(int[] numbers)
    {
        int result = 0;

        for (int i = 0; i < numbers.Length; i++)
        {
            result += numbers[i];
        }

        return result;
    }
}
```

Ref-22

```
class Program
{
    static void Main(string[] args)
    {
        DisplayHello("Allan");
    }
    static void DisplayHello(string name)
    {
        Console.WriteLine("Hello" + name);
    }
}
```