C# Cheat Sheet

12/24/2013

A cheat sheet to the C# language, ideal for newcomers to the language for more visit http://www.thecodingguys.net

KEEP IN TOUCH
## TABLE OF CONTENTS

### LICENSE

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

### LANGUAGE BASICS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>4</td>
</tr>
<tr>
<td>Variables</td>
<td>4</td>
</tr>
<tr>
<td>Syntax</td>
<td>4</td>
</tr>
<tr>
<td>Naming Rules</td>
<td>4</td>
</tr>
<tr>
<td>Example</td>
<td>4</td>
</tr>
<tr>
<td>Arrays</td>
<td>4</td>
</tr>
<tr>
<td>Syntax</td>
<td>4</td>
</tr>
<tr>
<td>Example</td>
<td>5</td>
</tr>
<tr>
<td>Strings</td>
<td>5</td>
</tr>
<tr>
<td>Concatenation</td>
<td>5</td>
</tr>
<tr>
<td>Example</td>
<td>5</td>
</tr>
<tr>
<td>New Line</td>
<td>5</td>
</tr>
<tr>
<td>Example</td>
<td>5</td>
</tr>
<tr>
<td>String.Format</td>
<td>5</td>
</tr>
<tr>
<td>Example</td>
<td>5</td>
</tr>
</tbody>
</table>

### CONDITIONAL STATEMENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>If statements</td>
<td>6</td>
</tr>
<tr>
<td>Syntax</td>
<td>6</td>
</tr>
<tr>
<td>Example</td>
<td>6</td>
</tr>
<tr>
<td>If Else Statements</td>
<td>6</td>
</tr>
<tr>
<td>Example</td>
<td>6</td>
</tr>
<tr>
<td>Switch Statement</td>
<td>6</td>
</tr>
<tr>
<td>Syntax</td>
<td>7</td>
</tr>
<tr>
<td>Example</td>
<td>7</td>
</tr>
</tbody>
</table>

### LOOPS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>While Loop</td>
<td>8</td>
</tr>
<tr>
<td>Syntax</td>
<td>8</td>
</tr>
<tr>
<td>Example</td>
<td>8</td>
</tr>
<tr>
<td>For Loop</td>
<td>8</td>
</tr>
<tr>
<td>Syntax</td>
<td>8</td>
</tr>
<tr>
<td>Example</td>
<td>9</td>
</tr>
</tbody>
</table>
ADVANCED – EXCEPTIONS, METHODS & CLASSES

Exceptions
Syntax 10
Example 10

Methods
Syntax 10
Example 11
Passing Parameters 11
Returning Data 11

Classes
Syntax 12
Example 12

SUMMARY
Why Not Give us a like? 12
This work is licensed under the creative commons Attribution-NonCommercial-NoDerivs 3.0 Unported

✖️ You may not alter, transform, or build upon this work.
✖️ You may not use this work for commercial purposes.
✔️ You are free to copy, distribute and transmit the work
LANGUAGE BASICS

INTRODUCTION

C# is a powerful Object Orientated language, for those coming from Java or C++ you should be able to pick up the syntax for C# quickly. A few points:

- The language is case-sensitive (So A and a are different)
- Lines terminate with semi-colons
- Code is put in code blocks {}
- Inline comments start with //
- Block comments start with /* */
- XML comments start with ///

VARIABLES

To declare a variable you specify the data type and variable name followed by a value.

SYNTAX

```
DataType variableName = value;
```

NAMING RULES

- Variables must start with underscore or letter
- Variables cannot contain spaces
- Variables can contain numbers
- Cannot contain symbols (accept underscore)

EXAMPLE

```
string Name = "thecodingguys";
int Year = 2013;
```

I will use these two variables throughout.

ARRAYS

Arrays are similar to variables, but can hold more than one value.

SYNTAX

```
DataType[] ArrayName = { Comma Separated Values } // Array of any size
DataType[] ArrayName = new DataType[3] {Command Separated Values } //Expects 3 values
```
**EXAMPLE**

```csharp
string[] MyGamesOf2013 = {"GTAV", "Battlefield3");
```

**STRINGS**

**CONCATENATION**

Concatenation is done through the + operator.

```csharp
Console.WriteLine("Hello " + "World");
```

**NEW LINE**

```csharp
Console.WriteLine("Hello \n" + "World");
```

**STRING.FORMAT**

Formats an object, you specify the formatting you wish to perform, the following formats an integer and displays the currency symbol.

```csharp
Console.WriteLine(string.Format("{0:C}", 5));
```

Depending on your computer's regional settings you will see £5.00 displayed (You'll see your countries currency symbol). The 0:C is the formatting we wish to do, in this case it means format the first parameter (0) and show a currency sign.
**CONDITIONAL STATEMENTS**

**IF STATEMENTS**

If statement is used to execute code based on a condition the condition must evaluate to true for the code to execute.

**Syntax**

```csharp
if (true)
{

}
```

**Example**

```csharp
if (Year > 2010)
{
    Console.WriteLine("Hello World!");
}
```

**IF ELSE STATEMENTS**

If a condition does not evaluate to true you can use an if else statement to execute other code.

**Example**

```csharp
if (Year > 2015)
{
    Console.WriteLine("Hello World!");
} else
{
    Console.WriteLine("Year is: " + Year);
}
```

**SWITCH STATEMENT**

Similar to the If else statement, however it has these benefits.
- Much easier to read and maintain
- Much cleaner than using nested if else
- It only evaluates one variable

**Syntax**

```csharp
switch (switch_on)
{
    default:
}
```

**Example**

```csharp
switch (Year)
{
    case 2013 :
        Console.WriteLine("It's 2013!");
        break;
    case 2012 :
        Console.WriteLine("It's 2012!");
        break;
    default :
        Console.WriteLine("It's " + Year + "!");
        break;
}
```

The break keyword is required as it prevents case falling.
**Loops**

**While Loop**
Continuously loops code until the condition becomes false.

**Syntax**
```csharp
while (true)
{
}
```

**Example**
```csharp
while (Year >= 2013)
{
    if (Year != 2100)
    {
        Console.WriteLine(Year++);
    }
    else
    {
        break;
    }
}
```
Make sure your condition evaluates to false at some point otherwise the loop is endless and it can result in errors.

**For Loop**
Similar to the While Loop, but you specify when the loop will end.

**Syntax**
```csharp
for (int i = 0; i < length; i++)
{
}
```
**Example**

```csharp
for (int i = 0; i <= 100; i++)
{
    Console.WriteLine(i);
}
```

This prints out 1 to 100. The expression can be easily broken down like this:

- `I = 0;`
- `I is less than or equal to 100? (True)`
- `Increment I by 1`

When `I` reaches 100 it will stop because `I` will no longer be less than 100 and will equal 100 so the condition is false.

**For Each**

The for each loop is used to loop around a collection. (Such as an array)

**Syntax**

```csharp
foreach (var item in collection)
{
}
```

**Example**

```csharp
foreach (string movie in MyMoviesOf2013)
{
    Console.WriteLine(movie);
}
```

Outputs all the elements in the `MyMoviesOf2013` array.
Exceptions

To catch any exceptions which are likely to occur you use a try catch block.

Syntax

```csharp
try
{

}
catch (Exception)
{

    throw;
}
```

Example

```csharp
try
{
    string result = "k";
    Console.WriteLine(Convert.ToInt32(result) + 10);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

The above code results in a format exception, because you can’t convert K to a number 😞

Methods

Syntax

```csharp
public void MethodName()
{
    //Does not return a value
}
```
public static void MethodName()
{
   //Does not return a value, the class does not need to be initialized
   //for this method to be used.
}

public static DataType MethodName()
{
   //Requires a value to be returned, class does not need to be
   initialized for this method to be used.
}

**EXAMPLE**

```csharp
public static void WelcomeUser()
{
   Console.WriteLine("Hello Guest!");
}
```

**Passing Parameters**

```csharp
public static void WelcomeUser(string Name)
{
   Console.WriteLine("Hello " + Name + "!");
}
```

Since both methods have the same name and different parameters (One takes no parameters
and the other one does) this is said to be an **overloaded method**.

**Returning Data**

```csharp
public static DateTime Tomorrow()
{
   return DateTime.Now.AddDays(1);
}
```

All the examples above are static, this allows me to use the methods without initializing the
class. You can read more about **Classes and Methods**. Also public methods are available outside
of the current class, private methods are only available in the current class.
**Classes**

**Syntax**

```csharp
Class MyClassName
{
}
```

**Example**

```csharp
class MyCar
{
   public void Manufacturer(string Manf)
   {
      Console.WriteLine(Manf);
   }
}
```

To use the method in the class, the class must be initialized first.

```csharp
MyCar NewCar = new MyCar();
NewCar.Manufacturer("Audi");
```

If the method was declared static I could simply do this:

```csharp
MyCar.Manufacturer("Audi");
```

Static methods are useful, make sure you are using the right design for your classes and methods. A good example is the Math class, to perform simple calculations you do not want to be initializing the class all the time, that’s why most methods are static.

**Summary**

This cheat sheet sums up the basics of C#, for experienced developers who are learning C# and users who already know programming basics, hopefully this document has helped you in some way, there was not much information or explaining but then again I’m assuming you’ve programmed before and know the basics 😊 For more visit http://www.thecodingguys.net

*Why Not Give us a like?*