# C# Reference

# Cheat Sheet

**Content**

1. C# Keywords
2. Contextual Keywords
3. C# Operators
4. C# Preprocessor Directives

**About the document**

This document **IS NOT:**

* For commercial purpose
* Microsoft official document
* Introduction to programming

This document IS:

* C# Cheat sheet
* Free to use and feel free to contribute and share

**C# Keywords**

**abstract**

Definition

*abstract keyword indicates that the abstracted element is not implemented. Elements can be classes, methods, properties, indexers, and events.*

Rules:

* A non-abstract class derived from an abstract class must include actual implementations of all inherited abstract elements.
* An abstract class cannot be instantiated.
* Abstract method declarations are only permitted in abstract classes.
* An abstract inherited property can be overridden in a derived class by including a property declaration that uses the override modifier.

**base**

Definition

*The base keyword is used in a derived class to access members of the base class.*

Example

This example shows how to specify the base-class constructor called when creating instances of a derived class.

public class BaseClass{

public BaseClass() { }

public BaseClass(int i) { }

}

public class DerivedClass : BaseClass {

public DerivedClass() : base() { } // Call to BaseClass()

public DerivedClass(int i) : base(i) { } // Call to BaseClass(int I )

}

}

**delegate**

Definition

*A delegate is a reference to a named or an anonymous method.*

Rules

* A delegate must have the same signature as the method that it’s point to.
* A delegate can be instantiated by associating it either a named or an anonymous method.

Example

delegate double MathAction(double num);

MathAction ma = Double;

ma += addition;

**enum**

Definition

*The enum keyword is used to declare a set of constants called the enumerator list.*

Rule:

* By default the first enumerator start with the value of 0.
* Enumerators can be initialized to override the default value.
* The approved types for an enum are byte,sbyte,short,ushort,int,mong,ulong

Example

enum Day : byte {Sat=1, Sun, Mon, Tue, Wed, Thu, Fri};

**extern**

Definition

*The extern keyword is used to declare a method that is implemented externally. A common use of the extern modifier is with DllImport attribute when you are using Interop services to call into unmanaged code. In this case, the method must also be declared as static.*

Example

[DllImport("avifil32.dll")]

private static extern void AVIFileInit();

**internal**

Definition

*The internal keyword is an access modifier, internal members are accessible only within the same assembly.*

**lock**

Definition

*The lock keyword marks a statement block as a critical section by obtaining the mutual-exclusion lock for a given object, executing a statement, and then releasing the lock.*

*The lock keyword ensures that one thread does not enter a critical section of code while another thread is in the critical section. If another thread tries to enter a locked code, it will wait, block, until the object is released.*

Example

private Object thisLock = new Object();

public void Withdraw() {

lock (thisLock) {

//Code here

} }

**out**

Definition

*The out keyword causes arguments to be passed by reference. it’s like the ref keyword , except that ref requires that the variable be initialized before it’s passed. to use an out parameter , both the method definition and the calling method must explicitly use the out keyword.*

Example

static void Method(out int i) { i = 44; }

static void Main ()

{

int value;

Method (out value);

Console.WriteLine (value); // value is now 44

}

**override**

Definition

*The override modifier is required to extend or modify the abstract or virtual implementation of an inherited method, property, indexer, or event.*

**params**

Definition

*With params keyword , you can specify a method parameter that takes a variable number of arguments.*

*you can send a comma-separated list of arguments of the type specified in the parameter declaration or an array of arguments of the specified type. no additional parameters are permitted after the params keyword in a method declaration, and only one params keyword is permitted in a method declaration.*

**sealed**

Definition

*When applied to a class, the sealed modifier prevents other classes from inheriting from it.*

**sizeof**

Definition

*Used to obtain the size in bytes for an unmanaged type. unmanaged type include the built-in types , enum types , pointer types, user-defined structs that don’t contain any fields or properties that are reference types.*

**typeof**

Definition

*Used to obtain the System.Type object for a type.*

Example

System.Type type = typeof(int);

**virtual**

Definition

*The virtual keyword is used to modify a method, property, indexer, or event declaration and allow for it to be overridden in a derived class.*

volatile

Definition

The volatile keyword indicates that a field might be modified by multiple threads that are executing at the same time. Fields that are marked volatile are not subject to compiler optimizations that assume access by a single thread.

**Contextual Keywords**

A contextual keyword is used to provide a specific meaning in the code, but it is not a reserved word in C#.

**partial**

Definition

*Partial type definitions allow for the definition of a class, struct, or interface to be split into multiple files.*

**C# Operators**

**x?.y**

Definition

*null conditional member access. Returns null if the left-hand operand is null.*

Example

int? length = customers?.Length; // null if customers is null

Customer first = customers?[0]; // null if customers is null

int? count = customers?[0]?.Orders?.Count(); // null if customers, the first customer, or Orders is null

**x ?? y**

Definition

*returns x if it is non-null; otherwise, returns y.*

Example

int a = 0;

int? b = null;

int c = b ?? a;

Console.WriteLine(c); // result 0

**t ? x : y**

Definition

if test t is true, then evaluate and return x; otherwise, evaluate and return y.

**C# Preprocessor Directives**

Preprocessor directives are commands that are interpreted by the compiler to do a conditional build process.

**#region , #endregion**

Definition

Lets you specify a block of code that you can expand or collapse when using the outlining feature of the Visual Studio Code Editor. In longer code files, it is convenient to be able to collapse or hide one or more regions so that you can focus on the part of the file that you are currently working on.

**C# Preprocessor Directives**