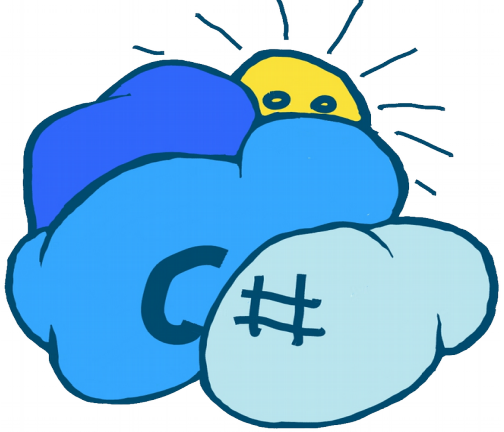
# Historyjka

**ECMA-334**

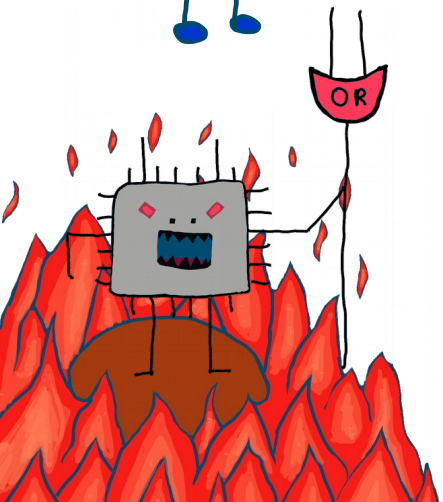
The C# Language Specification

**ECMA-335**

The Common Language Infrastructure





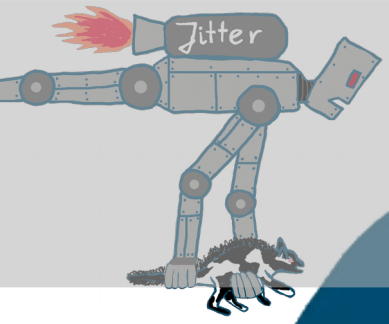



# IL – między piekłem a niebem.

**Krzysztof Owsiany**
MrDev.pl
After.conf
DevAdventCalendar.pl
Thenv

# Agenda

- Akronimy
- Jak to działa?
- Budowa pakietu
- Narzędzia
- Przykłady




# Akronimy

# CLI

**C**ommon **L**anguage **I**nfrastructure

# VES

**V**irtual **E**xecution **S**ystem

https://en.wikipedia.org/wiki/List_of_CLI_languages

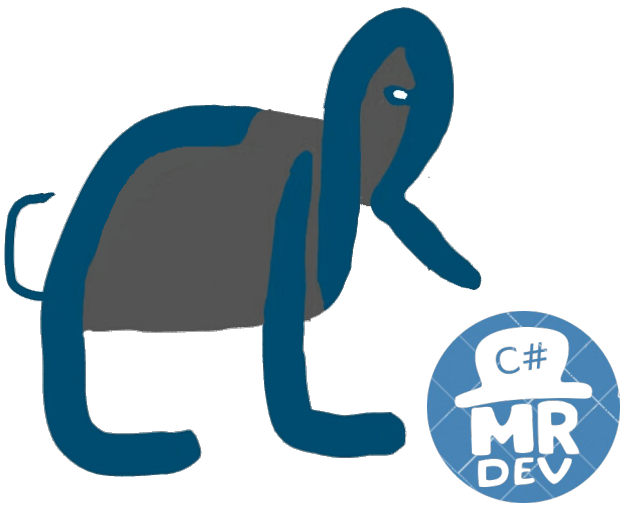
# CLR

**C**ommon **L**anguage **R**untime

# COREE

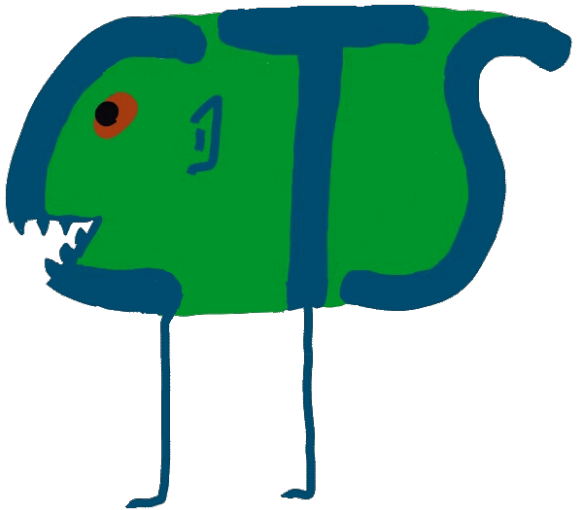
**C**ommon **O**bject **R**untime **E**xecution **E**ngine

mscoree.dll



# CTS

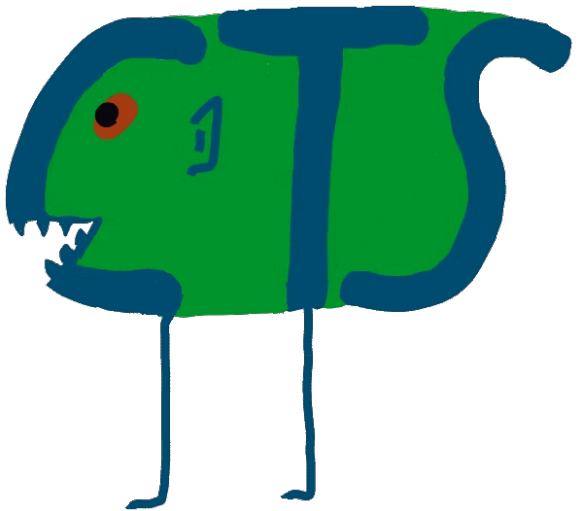
## Common Type System



**class**
**interface**
**struct**
**enum**
**delegate**

# CTS

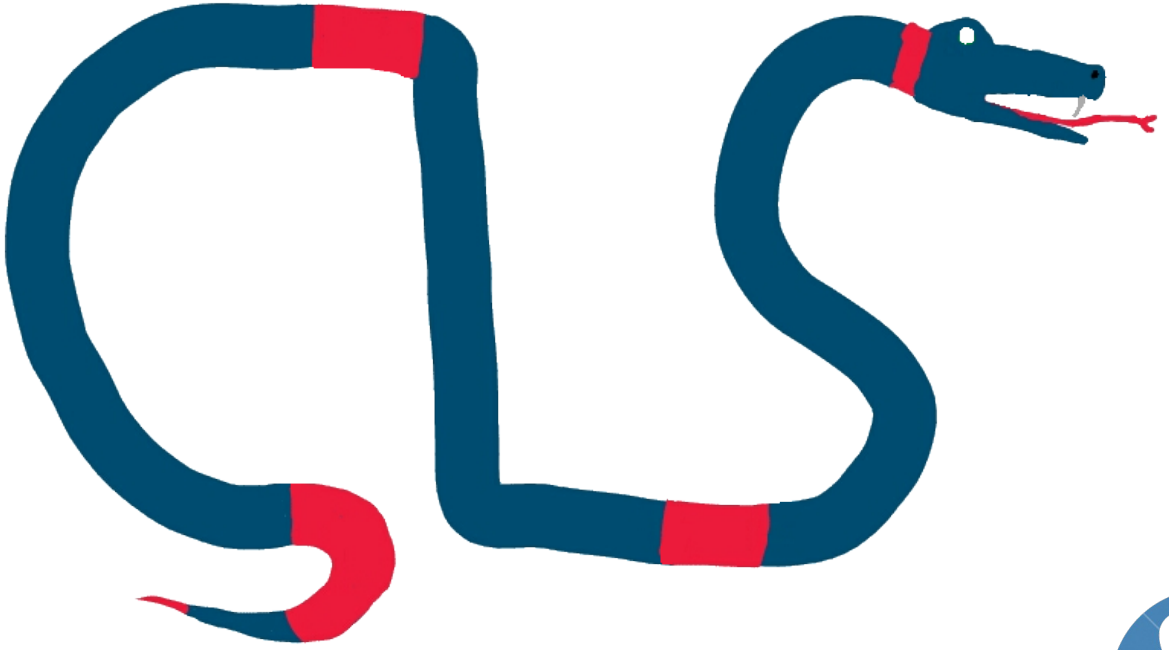
## Common Type System



**System.Byte/SByte**
**System.Int16/32/64**
**System.UInt16/32/64**
**System.Single/Double**
**System.Object**
**System.Char/String**
**System.Decimal**
**System.Boolean**

# CLS

## Common Language Specification




**IL**

**I**ntermediate **L**anguage

**CIL**

**C**ommon **I**ntermediate **L**anguage

**MSIL**

**M**icrosoft **I**ntermediate **L**anguage

https://en.wikipedia.org/wiki/List_of_CLI_languages

## CLI languages [edit]

### Current Languages [edit]

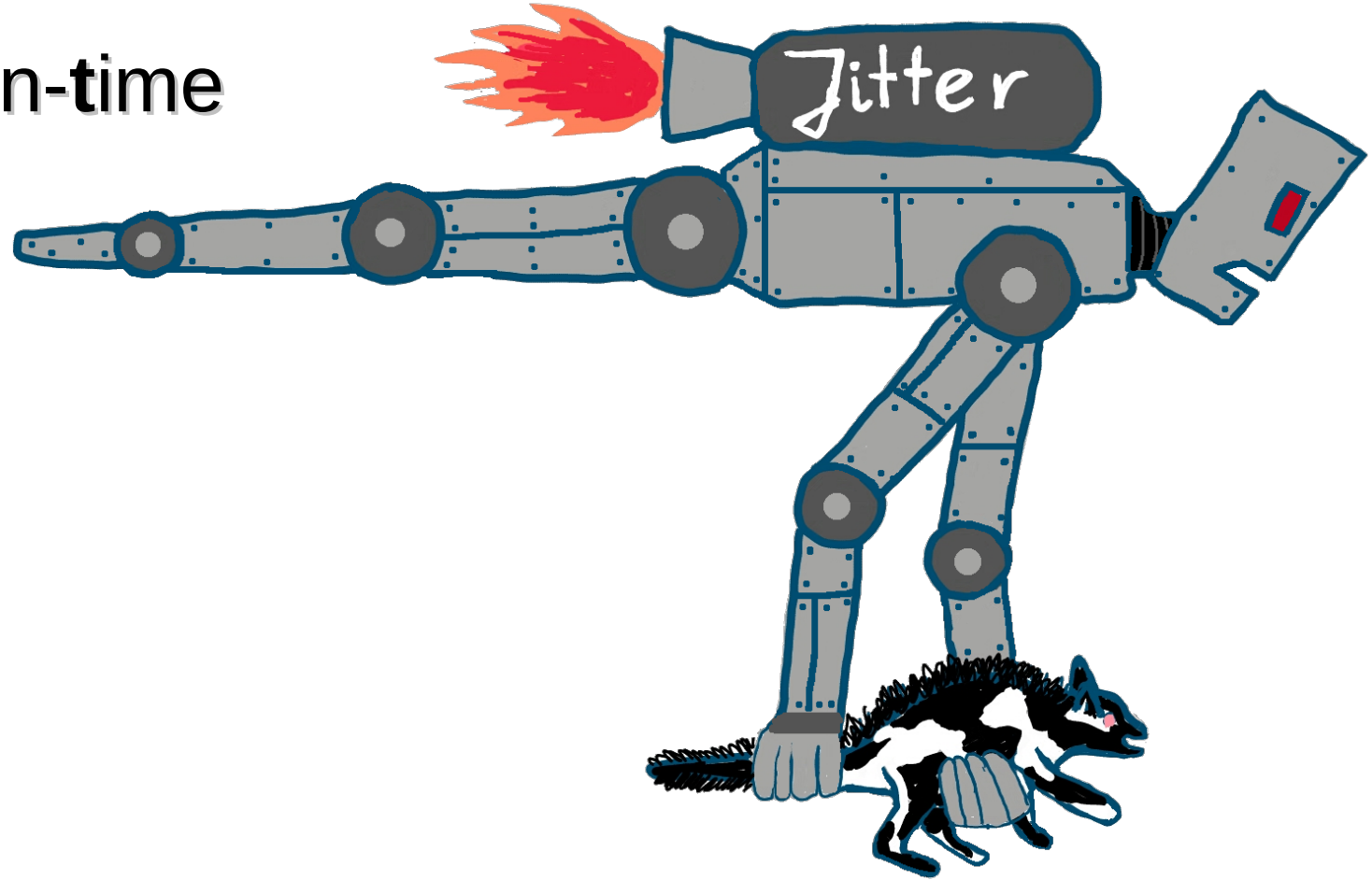
- C#: Most widely used CLI language [1], bearing similarities to Java, Object Pascal (Delphi) and C++. Implementations provided by .NET Framework, Portable.NET and Mono.
- C++/CLI: A version of C++ including extensions for using Common Language Runtime (CLR) objects. Implementation provided only by .NET Framework. Produces mixed-mode code that produces native code for C++ objects. The compiler is provided
- ClojureCLR: A native implementation of Clojure on the Common Language Runtime (CLR), the execution engine of Microsoft's .Net Framework.
- Cobra: A CLI language with static and dynamic typing, design by contract and built-in unit testing.
- Component Pascal: A CLI-compliant Oberon dialect. It is a strongly typed language in the heritage of Pascal and Modula-2 but with powerful object-oriented extensions.
- Eiffel: Purely object-oriented language, focused on software quality, includes integrated design by contract and multiple inheritance. CLI compliant.
- F#: A multi-paradigm CLI language supporting functional programming and imperative object-oriented programming disciplines. Variant of ML and is largely compatible with OCaml. The compiler is provided by Microsoft. The implementation provided b
- F* - A dependently typed language based on F#.
- Fantom - a language compiling to .NET and to the JVM
- IronPython: An open-source CLI implementation of Python, built on the Dynamic Language Runtime (DLR).
- IronScheme - a R6RS-compliant Scheme implementation built on the DLR
- JScript .NET: A CLI implementation of ECMAScript version 3, compatible with JScript. Contains extensions for static typing. Deprecated in favor of Managed JScript.
- L#: A CLI implementation of Lisp.
- Limnor Studio: Is a general-purpose codeless and visual programming system. The aim is to enable users to create computer software without coding in a textual programming language. It can be extended by software developers.
- Lisp# Un-Armed Bear Common Lisp (IKVM.NET port from Java)[2]
- Managed JScript: A CLI implementation of JScript built on the Dynamic Language Runtime (DLR). Conforms to ECMAScript version 3.
- Nemerle: A multi-paradigm language similar to C#, OCaml and Lisp.
- Oxygene: An Object Pascal-based CLI language.
- C#Prolog: A CLI implementation of Prolog from[3]
- Phalanger: An implementation of PHP with extensions for ASP.NET.
- Peachpie: A spiritual successor of Phalanger. An open-source implementation of PHP with extensions for ASP.NET_Core.[4]
- Phrogram: A custom CLI language for beginners and intermediate users produced by The Phrogram Company
- PowerBuilder: Can target CLI since version 11.1.
- Small Basic: A BASIC-derived programming language created by Microsoft for teaching programming. Supported releases target .NET Framework versions 3.5 and 4.5.
- Silverfrost FTN95: An implementation of Fortran 95.
- STARLIMS Scripting Language (SSL): A fully object-oriented BASIC like language implemented as server-side application language for the STARLIMS v10 / v11 enterprise software. SSL code gets compiled on the fly to MSIL for .NET CLR.
- Synergy DBL .NET: an object oriented CLI compliant implementation of DBL and DIBOL produced by Synergex.[5]
- Team Developer: SQLWindows Application Language (SAL) since Team Developer 6.0.
- Visual Basic .NET (VB.NET): A redesigned dialect of Visual Basic. Implementations provided by .NET Framework and Mono.
- Visual COBOL: an enhanced version of COBOL ported to the .NET Framework and to the JVM, produced by Micro Focus.[6]
- Visual RPG: a supercharged version of RPG ported to .NET by ASNA.
- Windows PowerShell: An object-oriented command-line shell. PowerShell can dynamically load .NET assemblies that were written in any CLI language. PowerShell itself uses a unique scripting syntax and uses curly-braces, similar to other C-based

### Abandoned or Deprecated Languages [edit]

- A#: CLI implementation of Ada.
- Axum: An actor model concurrent programming language.
- Boo: A statically typed CLI language, inspired by Python.
- Delphi.NET: CLI implementation of Delphi.
- Fortran for .NET[7]: CLI implementation of Fortran 77 and parts of Fortran 90/95 and Fortran 2003.
- GNAT for .NET: CLI implementation of Ada.
- GrGen.NET - a CLI language for graph rewriting
- IronLisp: A CLI implementation of Lisp. Deprecated in favor of IronScheme.
- IronRuby: An open-source CLI implementation of Ruby, built on the Dynamic Language Runtime (DLR).
- J#: A CLI-compliant implementation of Java. The compiler is provided by Microsoft. J# has been discontinued. The last version shipped with Visual Studio 2005, and was supported until 2015.
- Managed Extensions for C++: A version of C++ targeting the Common Language Runtime (CLR). Deprecated in favor of C++/CLI.
- Niecza - A CLI implementation of Perl 6.
- P#: A CLI implementation of Prolog.
- Prolog.NET: A CLI implementation of Prolog from[8][9]

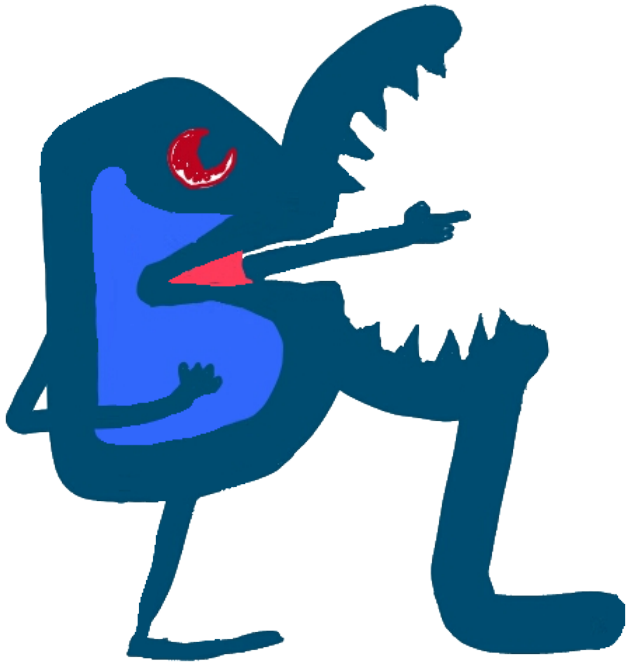
# JIT

## Just-in-time



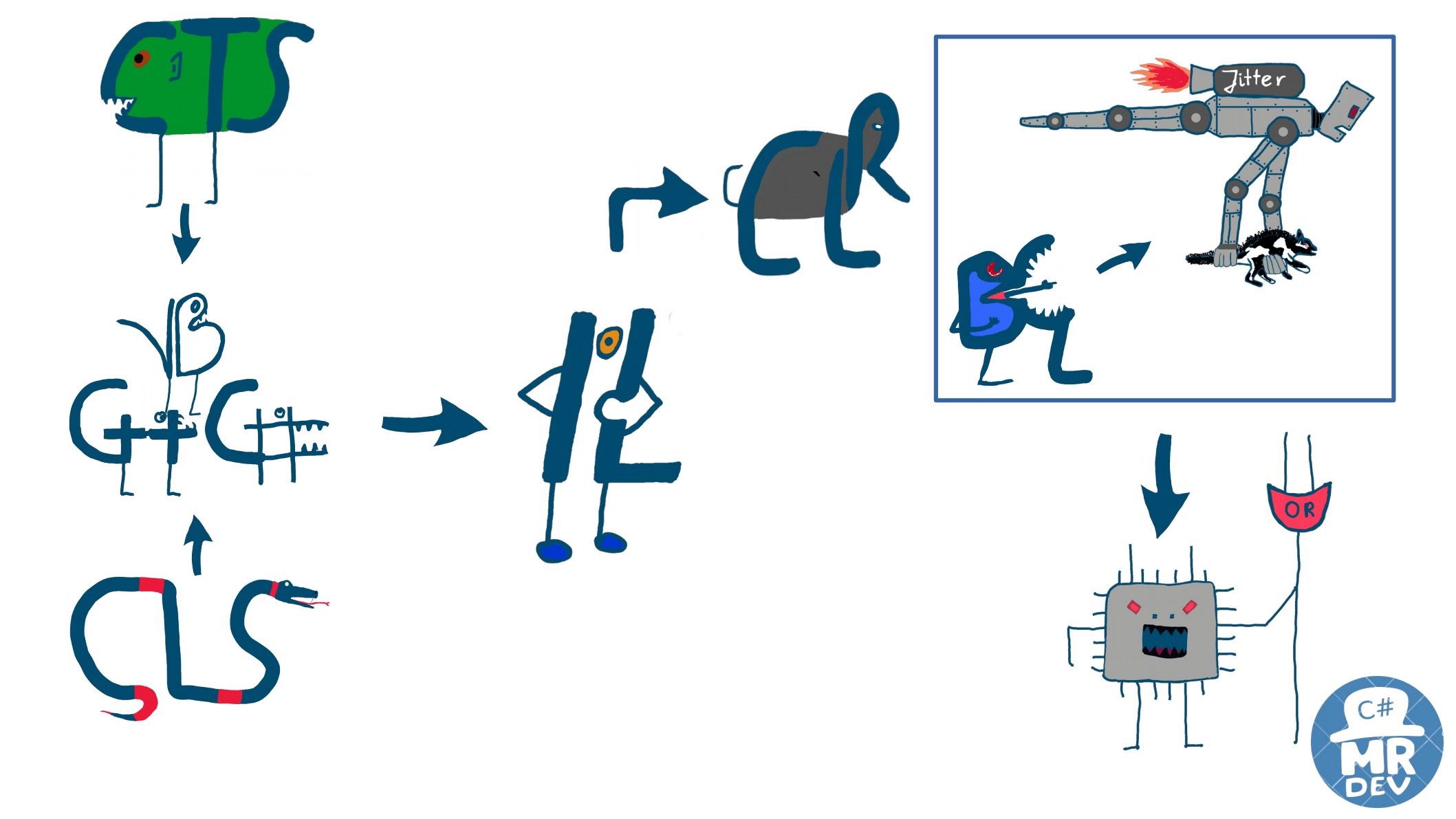
# BCL

## **B**asic **C**lass **L**ibrary

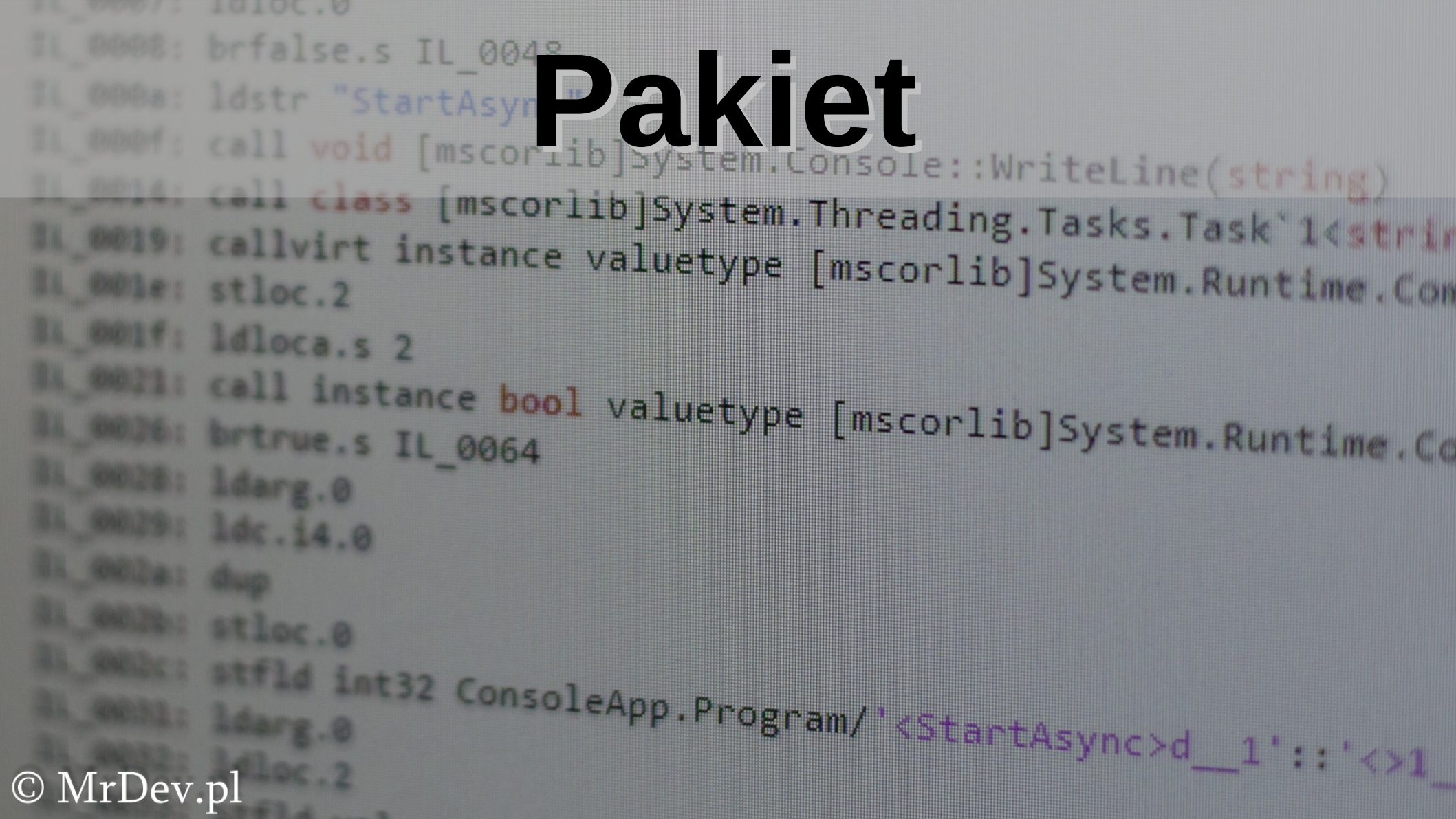


Obsługa wątków
Przetwarzanie grafiki
Operacje na plikach I/O
Komunikacja z peryferiami
Dostęp do bazy
Obsługa sieci
Zdalny dostęp
Zabezpieczenia

# Jak to działa?


Jitter
OR
C#
MR
DEV

# Pakiet



# Pakiet

- Nagłówki PE/COFF
- Nagłówek CLR
- Dane CLR
  - Metadane
  - Kod IL-a
  - Informacje o uchwyconych wyjątkach
  - Zasoby
- Dane i kod natywny

# Nagłówek CLR

- Wersja
- Kultura
- Klucz publiczny
- Lista zewnętrznych pakietów
- Informacje o aplikacji (AssemblyInfo.cs)

# Nagłówek CLR – pakiet

**.assembly** flaga nazwa
{
}

**Flagi:**
**<break>**
**cli**
**x86**
**ia64**
**amd64**
**arm**
**retargetable**
**windowsruntime**

# Nagłówek CLR – pakiet

```
.assembly NazwaPakietu
{
    .publickey = (00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00 )
    .hash algorithm 0x00008004
    .ver 2:0:0:0
    .locale „pl-PL”

    .custom instance void
    [mscorlib]System.Runtime.CompilerServices.RuntimeCompatibilityAttribute::.ctor()
}
```

# Nagłówek CLR – zewnętrzne pakiety

**.assembly extern** flaga nazwa
{
}

**Flagi:**
**<break>**
**cli**
**x86**
**ia64**
**amd64**
**arm**
**windowsruntime**

# Nagłówek CLR – zewnętrzne pakiety

```
.assembly extern NazwaWymaganegoPakietuZewnętrznego
{
        .ver 4:0:0:0
        .publickey = (B7 7A 5C 56 19 34 E0 89 )
        .publickeytoken = (B7 7A 5C 56 19 34 E0 89 )
        .locale „pl-PL”
        .hash = (B7 7A 5C 56 19 34 E0 89 )
}
```

# Nagłówek CLR

**.module** NazwaPakietu.**TYP**
**.subsystem** 0x0003
**.stackreserve** 0x00100000

# Dane CLR - metadane

- InteliSense
- Debugowanie
- Kompilator
- Przeglądanie obiektów
- Późne dowiązanie
- Mechanizm refleksji
- WCF
- Serializacji obiektów

# Dane CLR - metadane – tabele

0x00 - Module
0x01 - TypeRef
0x02 - TypeDef
0x03 - FieldPtr
0x04 - Field
0x05 - MethodPtr
0x06 - Method
0x07 - ParamPtr
0x08 - Param
0x09 - InterfaceImpl
0x0A - MemberRef
0x0B - Constant
0x0C - CustomAttribute
0x0D - FieldMarshal
0x0E - DeclSecurity
0x0F - ClassLayout
0x10 - FieldLayout
0x11 - StandAloneSig
0x12 - EventMap
0x13 - EventPtr
0x14 - Event
0x15 - PropertyMap
0x16 - PropertyPtr
0x17 - Property
0x18 - MethodSemantics
0x19 - MethodImpl

# Dane CLR - metadane – tabele

0x1A - ModuleRef
0x1B - TypeSpec
0x1C - ImplMap
0x1D - FieldRVA
0x1E - ENCLog
0x1F - ENCMap
0x20 - Assembly
0x21 - AssemblyProcessor
0x22 - AssemblyOS
0x23 - AssemblyRef
0x24 - AssemblyRefProcessor
0x25 - AssemblyRefOS
0x26 - File
0x27 - ExportedType
0x28 - ManifestResource
0x29 - NestedClass
0x2A - GenericParam
0x2B - MethodSpec
0x2C - GenericParamConstraint

# Dane CLR - metadane – token

1 bajt – identyfikator typu
3 bajty – RID (Record Identifier)

```
=============================================================
ScopeName : SimpleProgram.exe
MVID      : {6F0AFD4D-2E0C-4C1C-8192-33111F8EB084}
=============================================================
Global functions
-------------------------------------------------------
Global fields
-------------------------------------------------------
Global MemberRefs
-------------------------------------------------------
TypeDef #1 (02000002)
-------------------------------------------------------
        TypDefName: ConsoleApp.Program  (02000002)
        Flags     : [NotPublic] [AutoLayout] [Class] [AnsiClass] [BeforeFieldInit]  (00100000)
        Extends   : 01000001 [TypeRef] System.Object
        Method #1 (06000001) [ENTRYPOINT]
        -------------------------------------------------------
                MethodName: Main (06000001)
                Flags     : [Private] [Static] [HideBySig] [ReuseSlot]  (00000091)
                RVA       : 0x00002050
                ImplFlags : [IL] [Managed]  (00000000)
                CallCnvntn: [DEFAULT]
                ReturnType: Void
                1 Arguments
                        Argument #1:  SZArray String
                1 Parameters
                        (1) ParamToken : (08000001) Name : args flags: [none] (00000000)
```

# namespace

```
.namespace Examples
{
        .namespace SubExamples
        {

        }
}

.namespace Examples.SubExamples
{

}
```

# class

```
.namespace Examples {
    .class Example {
        .method public hidebysig specialname rtspecialname instance
            void .ctor() cli managed {}
    }
}
```

# class – metody

```
.namespace Examples {
    .class Example {
        .method private hidebysig instance int32 PrivateMethod() cli managed
        {}

        .method public hidebysig instance string PublicMethod() cli managed
        {}

        .method public hidebysig static void StaticMethod() cli managed
        {}
    }
}
```

# class – metody – dyrektywy

```
.maxstack 8
.locals init(int32 var1, int16 var2)
.entrypoint
```

# class – property

```
.namespace Examples {
    .class Example {
        .field private int32 exampleVar = int32(0)

        .property instance string TheString()
        {
            .get instance string Examples.Example.Get()
            .set instance string Examples.Example.Set(string)
        }

        .method public hidebysig instance string Get() cli managed {}

        .method public hidebysig instance Set(string 'value') cli managed {}
    }
}
```

# class – implements

```
.namespace Examples {
        .class public interface IExample
        {
        }

        .class Example implements Examples.IExample
        {
        }
}
```

# class – extends

```
.namespace Examples {
    .class public interface IExample
    {
    }

    .class public Example implements Examples.IExample
    {
    }

    .class Example2 extends Examples.Example
    {
    }
}
```

# class – extends + implements

```
.namespace Examples {
    .class public interface IExample {}
    .class public interface IExample2 implements Examples.IExample {}

    .class Example1
    {
    }

    .class Example2 extends Examples.Example1
    {
    }

    .class Example4
        extends Examples.Example2
        implements Examples.IExample {}
}
```

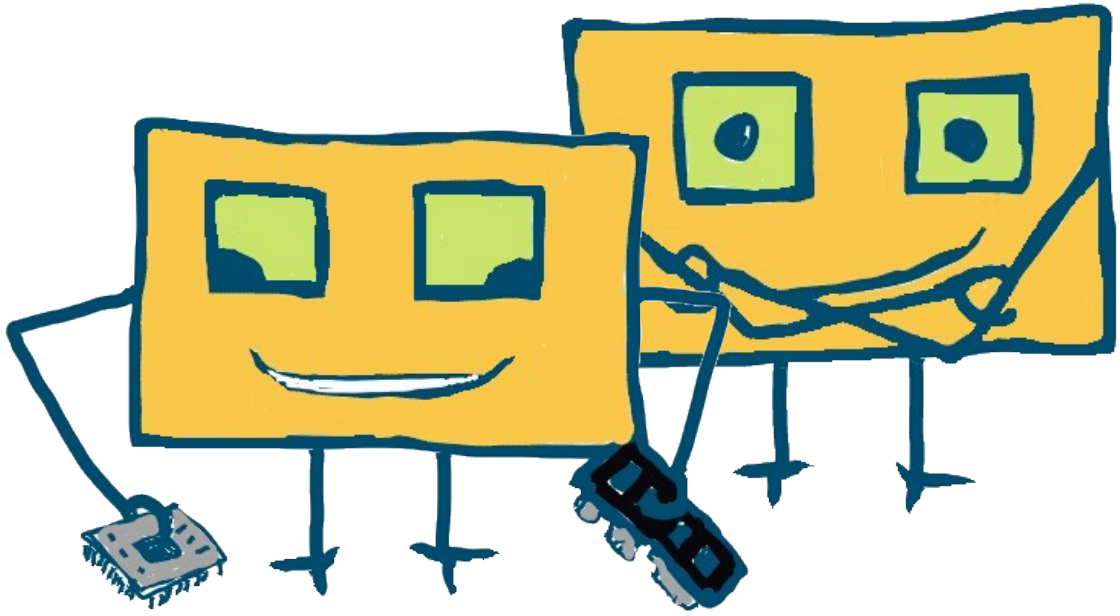
# struct

```
.namespace Examples {
    .class public sealed ExampleStruct
        extends [mscorlib]System.ValueType()
    {

    }

    .class public sealed value ExampleStruct2
    {

    }
}
```

# enum

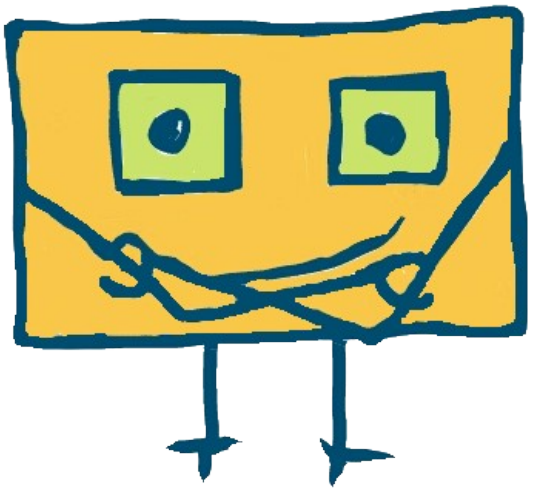
```
.namespace Examples {
    .class public sealed ExampleEnum extends [mscorlib]System.Enum()
    {
        .field public static literal valuetype
        Examples.ExampleEnum.Test = int32(1)
    }

    .class public sealed enum ExampleEnum2
    {

    }
}
```

# Kod operacyjny

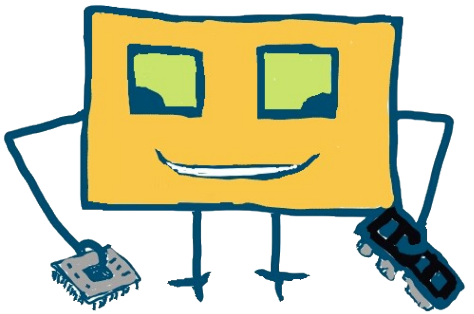


https://en.wikipedia.org/wiki/List_of_CIL_instructions

# Działania arytmetyczne

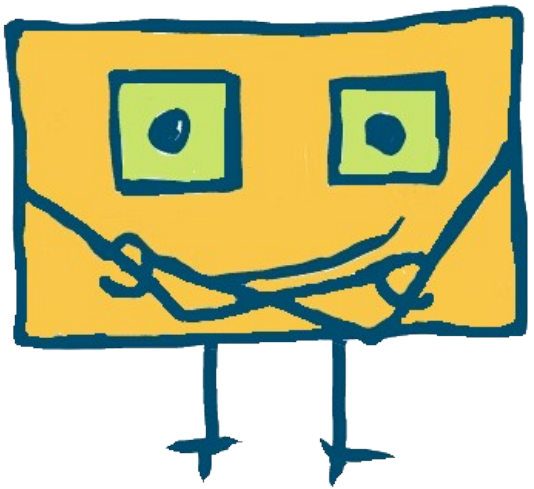


**add**
**sub**
**mul**
**div**
**rem**

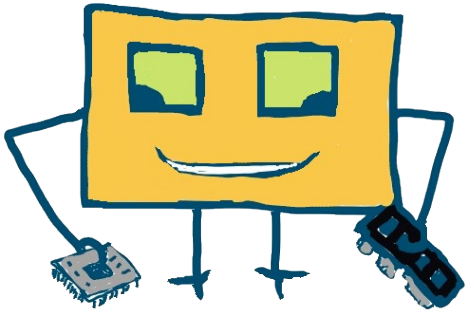


https://en.wikipedia.org/wiki/List_of_CIL_instructions

# Działania logiczne

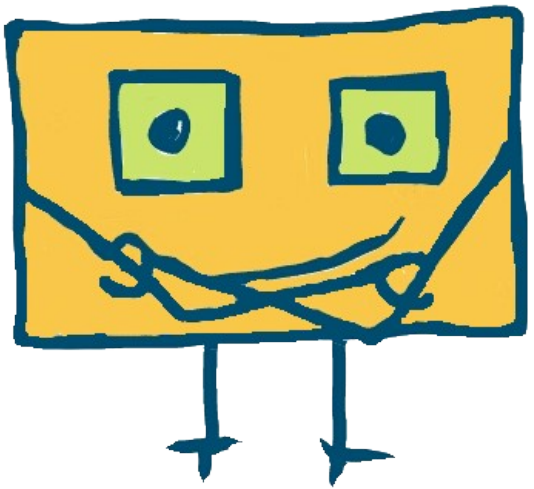


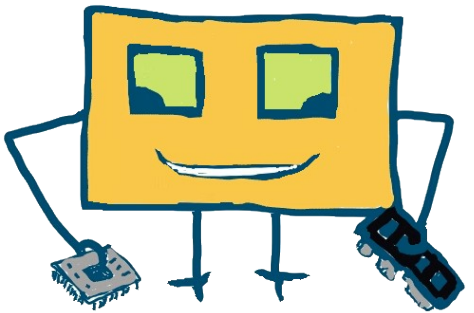
**and**
**or**
**not**
**xor**



https://en.wikipedia.org/wiki/List_of_CIL_instructions

# Konwersja typów

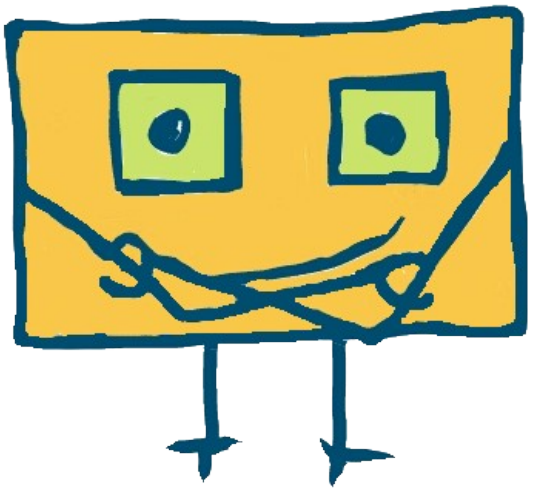




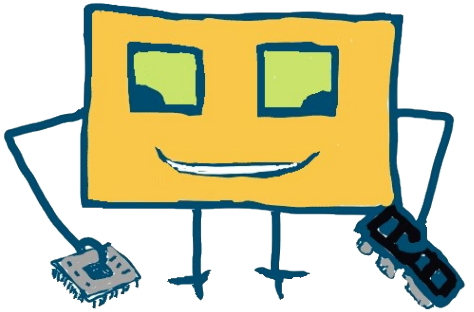
**box**
**unbox**

https://en.wikipedia.org/wiki/List_of_CIL_instructions

# Metody

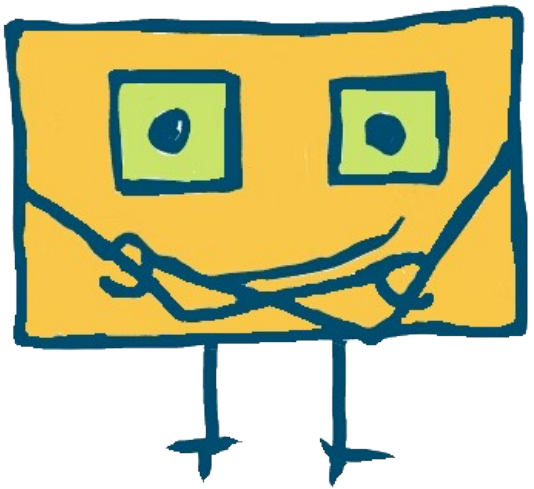


**call**
**ret**

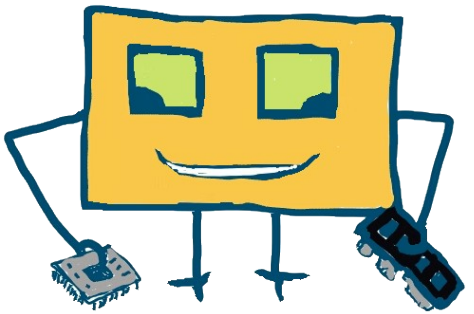


https://en.wikipedia.org/wiki/List_of_CIL_instructions

# Tworzenie

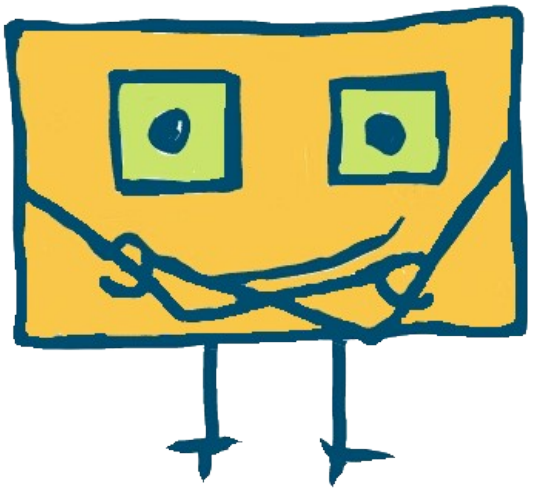


**newarr**
**newobj**

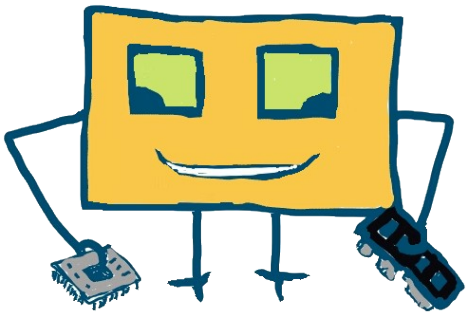


https://en.wikipedia.org/wiki/List_of_CIL_instructions

# Ładowanie na stos

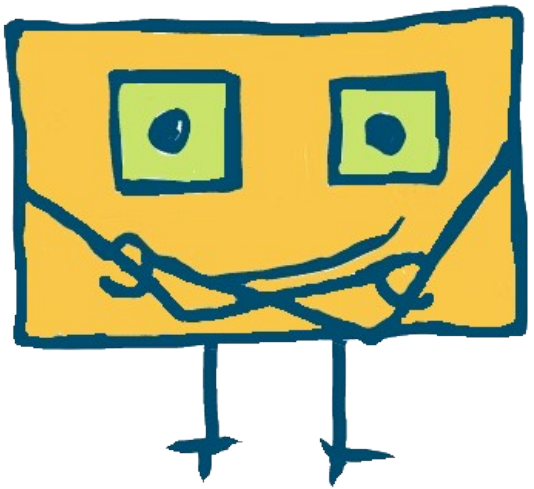


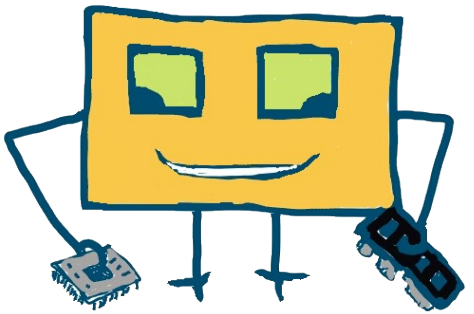
**ldarg**
**ldc**
**ldfld**
**ldloc**
**ldobj**
**ldstr**
**dup**



https://en.wikipedia.org/wiki/List_of_CIL_instructions

# Zdejmowanie ze stosu

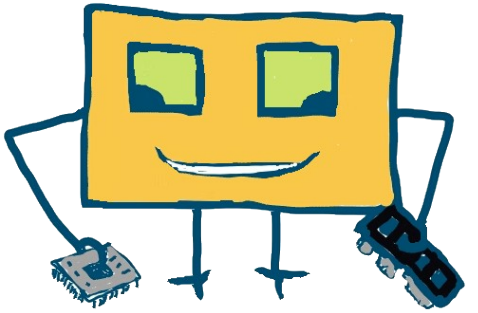


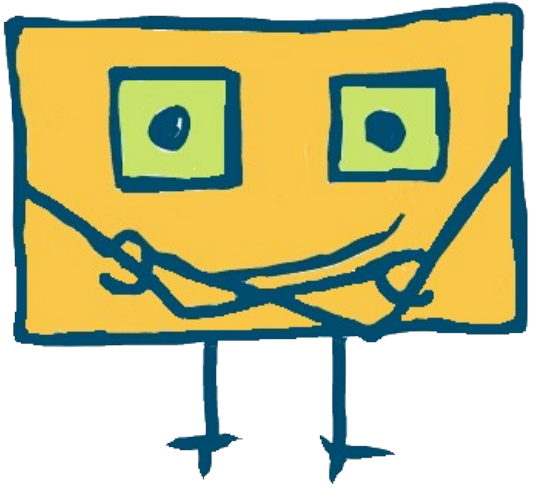


**pop**
**starg**
**stloc**
**stobj**
**stsfld**

https://en.wikipedia.org/wiki/List_of_CIL_instructions

# Porównanie na stosie

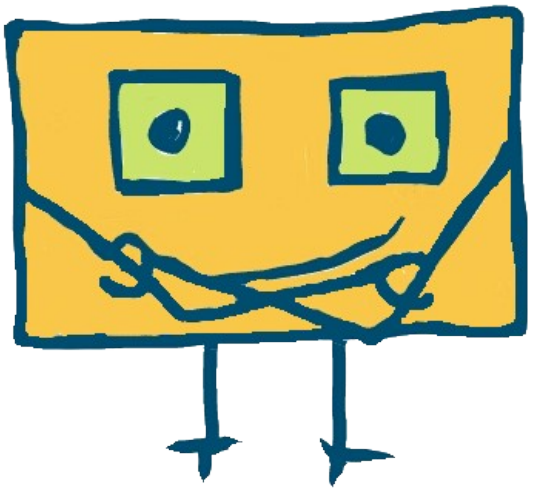




**ceq**
**cgt**
**clt**

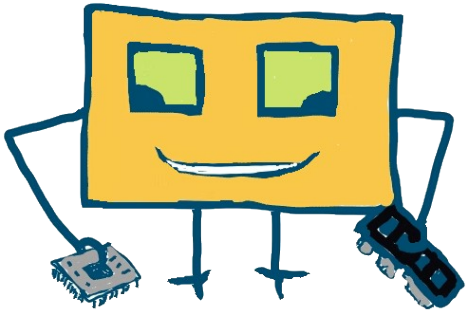
https://en.wikipedia.org/wiki/List_of_CIL_instructions

# Skoki



**br.s**
**beq**
**bgt**
**ble**
**blt**

**IL_####**



https://en.wikipedia.org/wiki/List_of_CIL_instructions

# Narzędzia

# Inżynieria wsteczna

## ildasm

Intermediate Language Dissasembler
(*C:\Program Files\Microsoft SDKs\Windows*)

## dumpbin

(*C:\Program Files (x86)\Microsoft Visual Studio 12.0*)

## Tune

https://github.com/kkokosa/Tune

## ILSpy

https://github.com/icsharpcode/ILSpy

# Kompilatory

**csc**

C Sharp Compiler

**vbc**

Visual Basic Compiler

**cl**

C++ Compiler

**ilasm**

Intermediate Language Assembler Compiler

# Pozostałe

## peverify

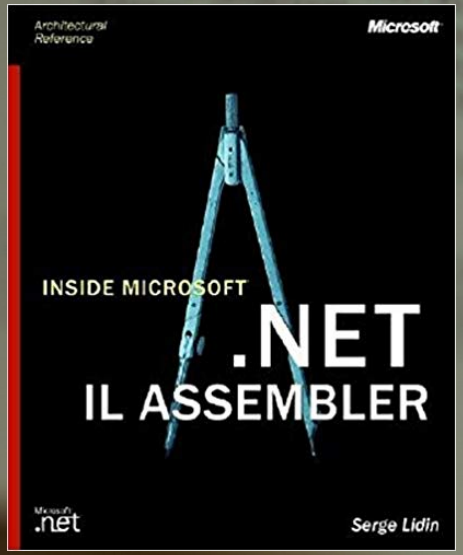
weryfikacja poprawności pakietu .NET

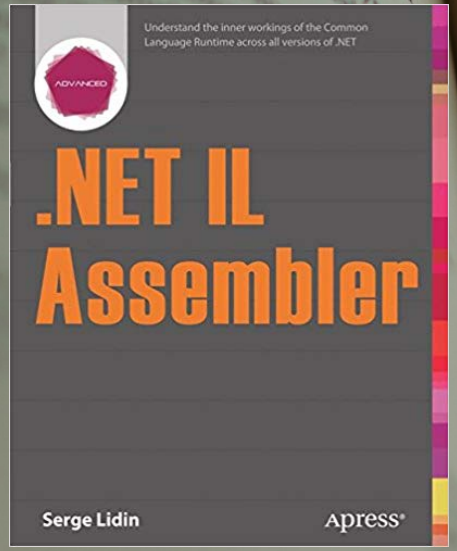
## ngen

pre-JIT, skraca czas uruchomienia

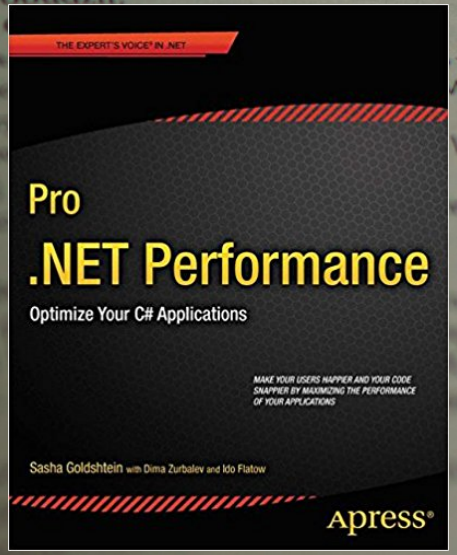
# Przykłady

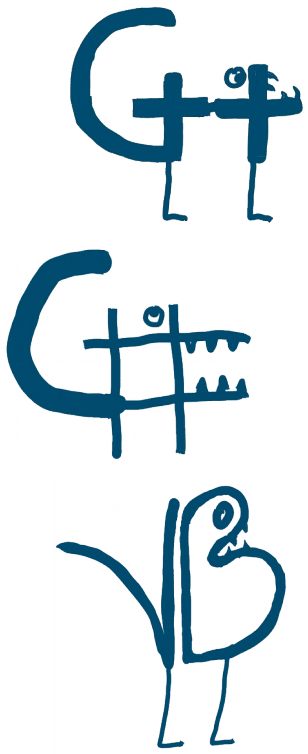
# Książki

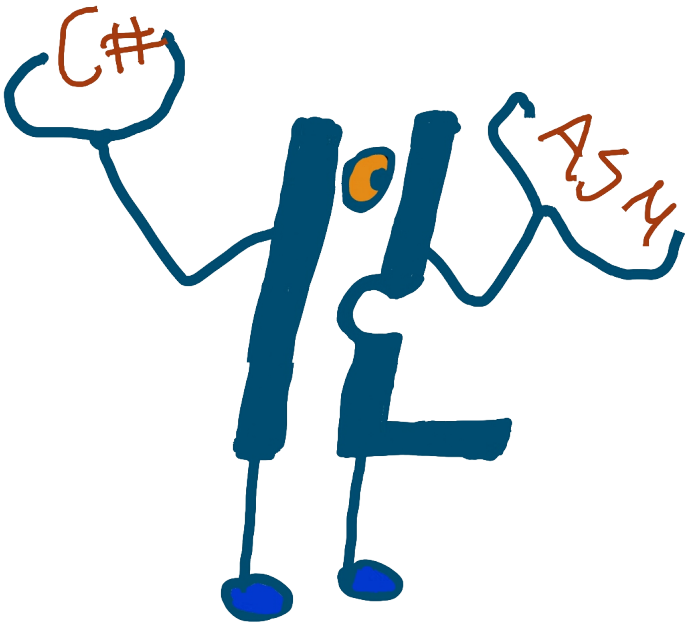


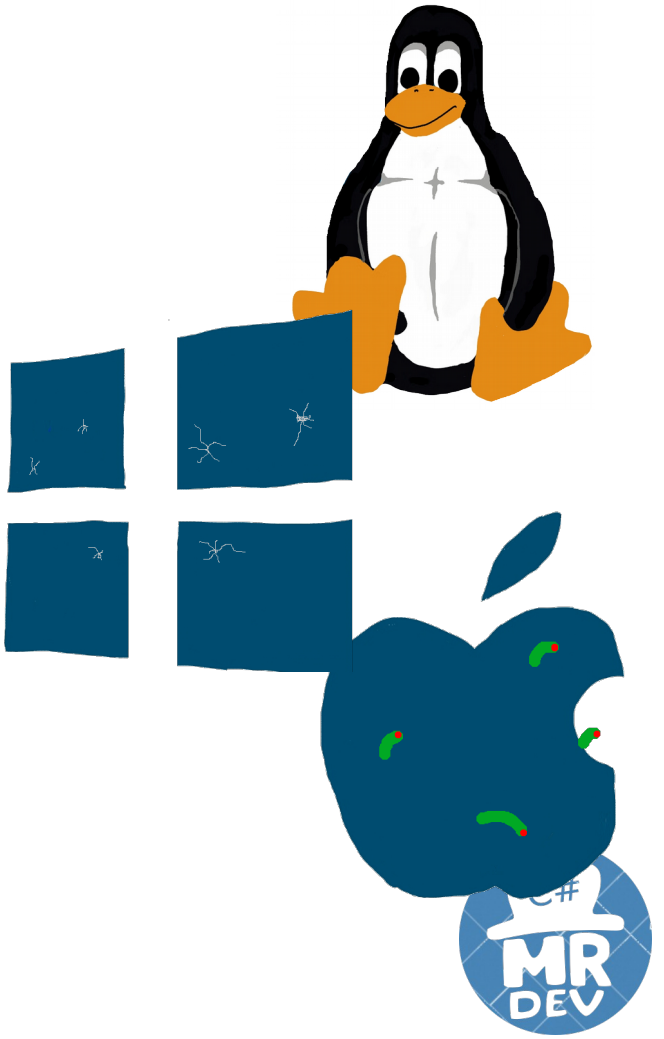










C#
ASY


C#
MR
DEV