Format générale

using System;

namespace YourNamespace

{

class YourMainClass

{

Type\_return NameMethod( arguments\_method ){

//Your mrthod starts here...

}

.

.

.

static void Main(string[] args)

{

//Your program starts here...

}

}

}

Start

**Start**: {using name\_space;} program

**name\_space** : System

**program** : namespace identifier{name\_space\_implement} | name\_space\_implement

**Name\_space\_implement** : [p\_p] *class* identifier{ class\_implement }

**p\_p** : private | public

**class\_implement**: {methode\_declaration} [main\_implement]

**main\_implement** : [public] static void Main(string[] identifier) methode\_body

**methode\_declaration** :

[p\_p] [static] return\_type methode\_name([formal\_parameter\_list]) methode\_body

**return\_type**: type | void

**methode\_name** : meth\_identifier

**formal\_parameter\_list**: fixed-parameters {,parameter-array} | parameter-array

**fixed-parameters**: fixed-parameter {, fixed-parameter}

**fixed-parameter** : type identifier

**parameter-array**: params array-type identifier

**methode\_body**: block | ;

Types

**type**: value-type [[]]

**value-type**: int | long |char | float | double | bool | string

**array-type**: value-type[]

**identifier** : letter {character}

Statement

**block**: { {statement} }

**statement**: declaration-statement | embedded-statement

**embedded-statement**: block | print; | statement-expression; | selection-statement | iteration-statement | jump-statement | try-statement

**print:** [System.]Console.Writeline({expression+} [expression])

**declaration-statement**: local-variable-declaration ; | local-constant-declaration ;

**local-variable-declaration**: type variable-declarator {, variable-declarator}

**variable-declarator**: identifier [= variable-initializer]

**variable-initializer**: expression

**local-constant-declaration**: const type constant-declarator {, constant-declarator}

**constant-declarator**: identifier = expression

**statement-expression**: invocation-expression | assignment

**selection-statement**: if ( boolean-expression ) embedded-statement [else embedded-statement] | switch ( identifier ) { { switch-section } }

**switch-section**: { switch-label } { statement } [break;]

**switch-label**: case expression : | default :

**iteration-statement**: while ( boolean-expression ) embedded-statement | do embedded-statement while ( boolean-expression ) ; | for ( [for-initializer] ; [boolean-expression] ; [for-iterator] ) embedded-statement | foreach ( type identifier in identifier ) embedded-statement

**for-initializer**: local-variable-declaration | assignment

**for-iterator**: statement-expression

**jump-statement**: break ; | continue ; | return [expression] ;

**boolean-expression**: expression

Expression

**expression**: conditional-or-expression

**assignment**: identifier assignment-body

**assignment-body: ++ | -- |** assignment-operator (type) expression

**assignment-operator**: = | += | -= | \*= | /= | %= | ^=

**conditional-or-expression**: conditional-and-expression { || conditional-and-expression}

**conditional-and-expression**: inclusive-or-expression { && inclusive-or-expression}

**inclusive-or-expression**: relational-expression {equality-operator relational-expression}

**equality-operator**: == | !=

**relational-expression**: additive-expression {relational-operator additive-expression }

**relational-operator**: < | > | <= | >=

**additive-expression**: multiplicative-expression {additive-operator multiplicative-expression}

**additive-operator**: + | -

**multiplicative-expression**: unary-expression { multiplicative-operator primary-expression }

**multiplicative-operator**: \* | / | %

**primary-expression**: identifier [[inum]] | (conditional-or-expression)| invocation-expression | inum | fnum | true | false | chaine | ‘character’ | array-creation-expression

**invocation-expression**: meth\_identifier ( [argument-list] )

**argument-list**: argument {, argument}

**argument:** expression

**array-creation-expression**: new value-type [array-length [array-initializer]

**array-length:** inum] | ]

**array-initializer**: {{expression,} [expression]}