A.2.10 Enums

enum-declaration:
attributes(optional)enum-modifiers(optional)enum identifier enum-base(optional)enum-body ;(optional)

enum-base:
: *integral-type*

enum-body:
{ *enum-member-declarations(optional)*}
{ *enum-member-declarations* , }

enum-modifiers:
enum-modifier
enum-modifiers enum-modifier

enum-modifier:
new
public
protected
internal
private

enum-member-declarations:
enum-member-declaration
enum-member-declarations , enum-member-declaration

enum-member-declaration:
attributes(optional)identifier
attributes(optional)identifier = constant-expression

argument-list:
    argument
    argument-list , argument

argument:
    expression
    ref variable-reference
    out variable-reference

primary-expression:
    primary-no-array-creation-expression
    array-creation-expression

primary-no-array-creation-expression:
    literal
    simple-name
    parenthesized-expression
    member-access
    invocation-expression
    element-access
    this-access
    base-access
    post-increment-expression
    post-decrement-expression
    object-creation-expression
    delegate-creation-expression
    typeof-expression
    checked-expression
    unchecked-expression

simple-name:
    identifier

parenthesized-expression:
    (expression)

member-access:
    primary-expression . identifier
    predefined-type . identifier
    predefined-type: one of
    bool byte char decimal double float int long
    object sbyte short string uint ulong ushort

invocation-expression:
    primary-expression (argument-list(optional))

element-access:
    primary-no-array-creation-expression [expression-list]

expression-list:
    expression
    expression-list , expression

this-access:
    this

base-access:
    base . *identifier*
    base [*expression-list*]

post-increment-expression:
    primary-expression ++

post-decrement-expression:
    primary-expression --

object-creation-expression:
    new type (argument-list(optional))
    type (argument-list(optional))

array-creation-expression:
new *non-array-type [expression-list] rank-specifiers(optional)array-initializer (optional)*
new *array-type array-initializer*

delegate-creation-expression:
expression

typeof-expression:
typeof (*type*)
typeof (void)

checked-expression:
checked (*expression*)
unchecked-expression:
unchecked (*expression*)

unary-expression:
primary-expression
monadic-expressions
\+ *monadic-expression*
\- *monadic-expression*
! *monadic-expression*
~ *monadic-expression*
* *monadic-expression*
≤ *monadic-expression*
≈ *monadic-expression*
≥ *monadic-expression*
≠ *monadic-expression*
⋁ *monadic-expression*
⋀ *monadic-expression*
× *monadic-expression*
÷ *monadic-expression*
∈ *monadic-expression*
monadic-expression
˜ *monadic-expression*
↑ *monadic-expression*
↓ *monadic-expression*
monadic-expression
○ *monadic-expression*
⌈ *monadic-expression*
⌊ *monadic-expression*
monadic-expression
monadic-expression
⊂ *monadic-expression*
⊃ *monadic-expression*
∣ *monadic-expression*
monadic-expression
≡ *monadic-expression*
monadic-expression
monadic-expression
monadic-expression
monadic-expression
⊖ *monadic-expression*
monadic-expression
monadic-expression
monadic-expression
monadic-expression
cast-expression

cast-expression:
(type) unary-expression

multiplicative-expression:
unary-expression
multiplicative-expression × *unary-expression*
multiplicative-expression ÷ *unary-expression*
multiplicative-expression % unary-expression
multiplicative-expression | *unary-expression*

additive-expression:

multiplicative-expression
additive-expression + *multiplicative-expression*
additive-expression – *multiplicative-expression*

shift-expression:
additive-expression
shift-expression << *additive-expression*
shift-expression >> *additive-expression*

relational-expression:
shift-expression
relational-expression < *shift-expression*
relational-expression > *shift-expression*
relational-expression <= *shift-expression*
relational-expression ≤*shift-expression*
relational-expression >= *shift-expression*
relational-expression ≤*shift-expression*
relational-expression is type
relational-expression as type

equality-expression:
relational-expression
equality-expression == *relational-expression*
equality-expression ≈*relational-expression*
equality-expression ≡ *relational-expression*
equality-expression != *relational-expression*
equality-expression ≠*relational-expression*

and-expression:
equality-expression
and-expression & *equality-expression*
and-expression ∧ *equality-expression*

inclusive-or-expression:
inclusive-or-expression | *exclusive-or-expression*

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

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

conditional-expression:
conditional-or-expression
conditional-or-expression then expression else expression

array-scalar-expression:
array-scalar-expression + *monadic-expression*
array-scalar-expression - *monadic-expression*
array-scalar-expression ! *monadic-expression*
array-scalar-expression ~ *monadic-expression*
array-scalar-expression * *monadic-expression*
array-scalar-expression ≤*monadic-expression*
array-scalar-expression ≈*monadic-expression*
array-scalar-expression ≥*monadic-expression*
array-scalar-expression ≠*monadic-expression*
array-scalar-expression ∨ *monadic-expression*
array-scalar-expression ∧ *monadic-expression*
array-scalar-expression × *monadic-expression*
array-scalar-expression ÷ *monadic-expression*
array-scalar-expression ∈ *monadic-expression*
array-scalar-expression *monadic-expression*
array-scalar-expression ~ *monadic-expression*
array-scalar-expression ↑ *monadic-expression*
array-scalar-expression ↓ *monadic-expression*
array-scalar-expression *monadic-expression*
array-scalar-expression ○ *monadic-expression*
array-scalar-expression ⌈ *monadic-expression*
array-scalar-expression ⌊ *monadic-expression*
array-scalar-expression *monadic-expression*

array-scalar-expression *monadic-expression*
array-scalar-expression ⊂ *monadic-expression*
array-scalar-expression ⊃ *monadic-expression*
array-scalar-expression | *monadic-expression*
array-scalar-expression *monadic-expression*
array-scalar-expression ≡ *monadic-expression*
array-scalar-expression *monadic-expression*
array-scalar-expression *monadic-expression*
array-scalar-expression *monadic-expression*
array-scalar-expression *monadic-expression*
array-scalar-expression ⊖ *monadic-expression*
array-scalar-expression *monadic-expression*
array-scalar-expression *monadic-expression*
array-scalar-expression *monadic-expression*
array-scalar-expression *monadic-expression*

assignment:
unary-expression assignment-operator expression

assignment-operator: one of
= += −= ×= ⍵= &= ⍲= ∧= <<= >>=
≤ ≈ ≥ ≠ ∨ ∧ ×= ÷= ∈ ⍴ ~
↑ ↓ ⌽ ⍉
⌈ ⌊ ⊂ ⊃ ⊥ ⊤ | ⍪
≡ ⍒ ⍋
⍫ ⍂ ⊖ ⍟ ⍱ ⍲ ⍷

expression:
conditional-expression
assignment

constant-expression:
expression

boolean-expression:
expression

A.2.5 Statements

statement:
    labeled-statement
    declaration-statement
    embedded-statement

embedded-statement:
    block
    empty-statement
    expression-statement
    selection-statement
    iteration-statement
    jump-statement
    try-statement
    checked-statement
    unchecked-statement
    lock-statement (not implemented)

block:
    { statement-list(optional)}

statement-list:
    statement
    statement-list statement

empty-statement:
    ; (optional)
    ◇ (optional)

labeled-statement:
    identifier : statement

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

local-variable-declaration:
    type local-variable-declarators

local-variable-declarators:
    local-variable-declarator
    local-variable-declarators , local-variable-declarator

local-variable-declarator:
    identifier (optional)
    identifier = local-variable-initializer

local-variable-initializer:
    expression
    array-initializer

local-constant-declaration:
    const *type constant-declarators*

constant-declarators:
    constant-declarator
    constant-declarators , constant-declarator

constant-declarator:
    identifier = constant-expression

expression-statement:
    statement-expression ;

statement-expression:
    invocation-expression
    object-creation-expression
    assignment
    post-increment-expression
    post-decrement-expression

pre-increment-expression
pre-decrement-expression

selection-statement:
if-statement
switch-statement

if-statement:
if *(boolean-expression*) *embedded-statement*
if (*boolean-expression*) *embedded-statement else embedded-statement*

boolean-expression:
expression

switch-statement:
switch (*expression*) *switch-block*

switch-block:
{ *switch-sections(optional)}*

switch-sections:
switch-section
switch-sections switch-section

switch-section:
switch-labels statement-list

switch-labels:
switch-label
switch-labels switch-label

switch-label:
case *expression* :
default :

iteration-statement:
while-statement
do-statement
for-statement
foreach-statement

while-statement:
while (*boolean-expression*) *embedded-statement*

do-statement:
do *embedded-statement* while (*boolean-expression*) ; *(optional)*

for-statement:
for (*for-initializer(optional); for-condition(optional*; *for-iterator(optional)) embedded-statement*

for-initializer:
local-variable-declaration
statement-expression-list

for-condition:
boolean-expression

for-iterator:
statement-expression-list

statement-expression-list:
statement-expression
statement-expression-list , statement-expression

foreach-statement:
foreach (*type(optional) identifier in expression*) *embedded-statement*

jump-statement:
break-statement
continue-statement
goto-statement

return-statement
throw-statement

break-statement:
break

continue-statement:
continue

goto-statement:
goto *identifier*

return-statement:
return *expression(optional)*;

throw-statement:
throw *expression(optional)*

try-statement:
try *block catch-clauses*
try *block finally-clause*
try *block catch-clauses finally-clause*

catch-clauses:
specific-catch-clauses general-catch-clauseopt
specific-catch-clauses(optional)general-catch-clause

specific-catch-clauses:
specific-catch-clause
specific-catch-clauses specific-catch-clause

specific-catch-clause:
catch (*class-type identifier(optional)*) *block*

general-catch-clause:
catch *block*

finally-clause:
finally *block*

checked-statement:
checked *block*

unchecked-statement:
unchecked *block*

lock-statement: (not implemented)
lock (*expression*) *embedded-statement*

resource-acquisition:
local-variable-declaration
expression

pp-directive:
    pp-declaration (not implemented)
    pp-conditional (not implemented)
    pp-line (not implemented)
    pp-diagnostic (not implemented)
    pp-region

pp-new-line:
    whitespace(optional)single-line-comment(optional)new-line

skipped-characters:
    whitespace(optional)not-number-sign input-characters(optional)

not-number-sign:
    Any input-character except #

pp-region:
    pp-start-region conditional-section(optional)pp-end-region

pp-start-region:
    whitespace(optional)# whitespace(optional)region pp-message

pp-end-region:
    whitespace(optional)# whitespace(optional)endregion pp-message

whitespace:
Any character with Unicode class Zs
Horizontal tab character (U+0009)
Vertical tab character (U+000B)
Form feed character (U+000C)

comment:
    single-line-comment
    delimited-comment

single-line-comment:
    // input-characters(optional)
       input-characters(optional)

input-characters:
    input-character
    input-characters input-character

input-character:
    Any Unicode character except a new-line-character

new-line-character:
    Carriage return character (U+000D)
    Line feed character (U+000A)

delimited-comment:
    / delimited-comment-characters (optional) /

delimited-comment-characters:
    delimited-comment-character
    delimited-comment-characters delimited-comment-character

delimited-comment-character:
    not-lamp
       not-slash

not-lamp:
    Any Unicode character except

not-slash:
    Any Unicode character except /

unicode-escape-sequence:
\u hex-digit hex-digit hex-digit hex-digit
\U hex-digit hex-digit hex-digit hex-digit hex-digit hex-digit hex-digit hex-digit

identifier:
    available-identifier
    @ identifier-or-keyword - not implemented (planned)

available-identifier:
    An identifier-or-keyword that is not a keyword

identifier-or-keyword:
    identifier-start-character identifier-part-characters(optional)

identifier-start-character:
    letter-character
    _ (the underscore character U+005F)

identifier-part-characters:
    identifier-part-character
    identifier-part-characters identifier-part-character

identifier-part-character:
    letter-character
    decimal-digit-character
    connecting-character
    combining-character
    formatting-character

letter-character:
    A Unicode character of classes Lu, Ll, Lt, Lm, Lo, or Nl
    A unicode-escape-sequence representing a character of classes Lu, Ll, Lt, Lm, Lo, or Nl

combining-character:
    A Unicode character of classes Mn or Mc
    A unicode-escape-sequence representing a character of classes Mn or Mc

decimal-digit-character:
    A Unicode character of the class Nd
    A unicode-escape-sequence representing a character of the class Nd

connecting-character:
    A Unicode character of the class Pc
    A unicode-escape-sequence representing a character of the class Pc

formatting-character:
    A Unicode character of the class Cf
    A unicode-escape-sequence representing a character of the class Cf

keyword: one of

abstract	as	base	bool	break	byte	case
catch	char	checked	class	const	continue	decimal
default	delegate	do	double	else	enum	event
explicit	extern	false	finally	fixed	float	for
foreach	goto	if	implicit	in	int	interface
internal	is	lock	long	namespace	new	null
object	operator	out	override	params	private	protected
public	readonly	ref	return	sbyte	sealed	short
sizeof	stackalloc	static	string	struct	switch	this
throw	true	try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	void	volatile	while
and	function	elseif	then	eval	global	step
not	or	nop	print	yield	repeat	until
definition	classtype	classop	optype	_arglist	_argnames	
referencebyname	referencebyfile					

A.1.8 Literals

literal:
    boolean-literal
    integer-literal
    real-literal
    character-literal
    string-literal
    null-literal
    type-literal

boolean-literal:
    true
    false

integer-literal:
    decimal-integer-literal
    hexadecimal-integer-literal

decimal-integer-literal:
    decimal-digits integer-type-suffix(optional)

decimal-digits:
    decimal-digit
    decimal-digits decimal-digit

decimal-digit: one of
    0 1 2 3 4 5 6 7 8 9

integer-type-suffix: one of
    U u L l UL Ul uL ul LU Lu lU lu

hexadecimal-integer-literal:
    0x hex-digits integer-type-suffix(optional)
    0X hex-digits integer-type-suffix(optional)

hex-digits:
    hex-digit
    hex-digits hex-digit

hex-digit: one of
    0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f

octal-integer-literal:
    0c octal-digits integer-type-suffix(optional)
    0C octal-digits integer-type-suffix(optional)

octal-digits:
    octal-digit
    octal-digits octal-digit

octal-digit: one of
    0 1 2 3 4 5 6 7

real-literal:
    decimal-digits . decimal-digits exponent-part(optional)real-type-suffix (optional)
    . decimal-digits exponent-part(optional)real-type-suffix (optional)
    decimal-digits exponent-part real-type-suffix (optional)
    decimal-digits real-type-suffix

exponent-part:
    e sign(optional)decimal-digits
    E sign(optional)decimal-digits

sign: one of
    + -¯

real-type-suffix: one of
    F f D d M m

character-literal:

'c'
' character '

character:
single-character
simple-escape-sequence
hexadecimal-escape-sequence
unicode-escape-sequence

single-character:
Any character except ' (U+0027), \ (U+005C), and new-line-character

simple-escape-sequence: one of
\' \" \\ \0 \a \b \f \n \r \t \v

hexadecimal-escape-sequence:
\x hex-digit hex-digit(optional)hex-digit(optional)hex-digit(optional)

characters:
array-characters
array-simple-escape-sequences
array-hexadecimal-escape-sequences
array-unicode-escape-sequences

array-character:
Any characters except ' (U+0027), \ (U+005C), and new-line-character

array-simple-escape-sequences: any of
\' \" \\ \0 \a \b \f \n \r \t \v

array-hexadecimal-escape-sequence:
\x hex-digit hex-digit(optional)hex-digit(optional)hex-digit(optional)

verbatim-characters:
@' verbatim -array-literal-characters(optional)'

verbatim-array-literal-characters:
verbatim-array-literal-character
verbatim-array-literal-characters
verbatim-array-literal-character

verbatim-array-literal-character:
single-verbatim-array-literal-character
quote-escape-sequence

single-verbatim-array-literal-character:
any character except '

string-literal:
regular-string-literal
verbatim-string-literal

regular-string-literal:
" regular-string-literal-characters(optional)"

regular-string-literal-characters:
regular-string-literal-character
regular-string-literal-characters
regular-string-literal-character

regular-string-literal-character:
single-regular-string-literal-character
simple-escape-sequence
hexadecimal-escape-sequence
unicode-escape-sequence

single-regular-string-literal-character:
Any character except " (U+0022), \ (U+005C), and new-line-character

verbatim-string-literal:

@" verbatim -string-literal-characters(optional)"

verbatim-string-literal-characters:
verbatim-string-literal-character
verbatim-string-literal-characters
verbatim-string-literal-character

verbatim-string-literal-character:
single-verbatim-string-literal-character
quote-escape-sequence

single-verbatim-string-literal-character:
any character except "

quote-escape-sequence:
""

null-literal:
null

operator-or-punctuator: one of

```
{ }  [ ]  ( )  .  ,  :  ;
+  −  ⋆  /  ⌿  &  ⍭  ^     ~                    !
=  <  >  ?  ++  −−  &&  ⍭⍭  <<  >>
==   =  <=  >=  +=  −=  ⋆=  /⃠=  ⌿⃠=  &=
⍭ = ^= <<= >>= −>
⋄ ¨ ¯ ≤ ≈ ≥ ≠ ∨ ∧ × ÷ ∈ ⍴ ∼ ↑ ↓ ⍳ ○ ⋆ ← →
⌈ ⌊ ∫ ∇ ∆ ∘ ⍁ ⍋ ⍒ ⊂ ⊃ ⊥ ⊤ ⍭ ⍀ ⌿ ≡ ⍤ ⍣ ⍫
⍬ ⍉ ⌽ ⊖ ⍟ ⍱ ⍲ ⍷ ⍺ ⍵
```

input:
    input-section (optional)

input-section:
    input-section-part
    input-section input-section-part

input-section-part:
    input-elements (optional)new-line
    pp-directive

input-elements:
    input-element
    input-elements input-element

input-element:
    whitespace
    comment
    token

namespace-name:
    namespace-or-type-name

type-name:
    namespace-or-type-name

namespace-or-type-name:
    identifier
    namespace-or-type-name . identifier

A.2.11 Attributes

global-attributes:
    global-attribute-sections

global-attribute-sections:
    global-attribute-section
    global-attribute-sections global-attribute-section

global-attribute-section:
    [*global-attribute-target-specifier attribute-list*]
    [*global-attribute-target-specifier attribute-list ,*]

global-attribute-target-specifier:
    global-attribute-target :
    global-attribute-target:
    assembly
    module

attributes:
    attribute-sections

attribute-sections:
    attribute-section
    attribute-sections attribute-section

attribute-section:
    [*attribute-target-specifier(optional)attribute-list*]
    [*attribute-target-specifier(optional)attribute-list ,*]

attribute-target-specifier:
    attribute-target :

attribute-target:
    field
    event (not implemented)
    method
    param (not implemented)
    property
    return (not implemented)
    type

attribute-list:
    attribute
    attribute-list , attribute

attribute:
    attribute-name attribute-arguments(optional)

attribute-name:
    type-name

attribute-arguments:
    (positional-argument-list(optional))
    (positional-argument-list , named-argument-list)
    (named-argument-list)

positional-argument-list:
    positional-argument
    positional-argument-list , positional-argument

positional-argument:
    attribute-argument-expression

named-argument-list:
    named-argument
    named-argument-list , named-argument

named-argument:
    identifier = attribute-argument-expression
    attribute-argument-expression:

expression

A.2.2 Types

type:
value-type
reference-type

value-type:
enum-type

struct-type:
type-name
simple-type

simple-type:
numeric-type
bool

numeric-type:
integral-type
floating-point-type
decimal
integral-type:
sbyte
byte
short
ushort
int
uint
long
ulong
char

floating-point-type:
float
double

enum-type:
type-name

reference-type:
class-type
interface-type
array-type
delegate-type

class-type:
type-name
object
string

interface-type:
type-name

array-type:
non-array-type rank-specifiers

non-array-type:
type

rank-specifiers:
rank-specifier
rank-specifiers rank-specifier

rank-specifier:
[dim-separators(optional)]

dim-separators:
,
dim-separators ,
;
dim-separators ;

delegate-type:
    type-name

variable-reference:
expression

compilation-unit:
using-directives(optional)global-attributes(optional)namespace-member-declarations(optional)
reference-directives(optional)namespace-member-declarations(optional)

namespace-declaration:
namespace *qualified-identifier namespace-body* ;(optional)

qualified-identifier:
identifier
qualified-identifier . identifier

namespace-body:
{ *reference-directives(optional)namespace-member-declarations(optional)*}
{ *using-directives(optional)namespace-member-declarations(optional)*}

reference-directives:
reference-directive
referencebyfile filename
referencebyfile using-directive

using-directives:
using-directive
using-directives using-directive

using-directive:
using-alias-directive
using-namespace-directive

using-alias-directive:
using *identifier = namespace-or-type-name* ;
using *namespace-or-type-name* as *identifier (depricated)*

using-namespace-directive:
using *namespace-name* ;

namespace-member-declarations:
namespace-member-declaration
namespace-member-declarations namespace-member-declaration

namespace-member-declaration:
namespace-declaration
type-declaration

type-declaration:
class-declaration
interface-declaration
enum-declaration

A.2.7 Classes

class-declaration:
attributes(optional)class-modifiers(optional)class identifier class-base(optional)class-body ;(optional)

class-modifiers:
class-modifier
class-modifiers class-modifier

class-modifier:
new
public
protected
internal
private
abstract
sealed

class-base:
: *class-type*
: *interface-type-list*
: *class-type , interface-type-list*

interface-type-list:
interface-type
interface-type-list , interface-type

class-body:
{ *class-member-declarations(optional)*}

class-member-declarations:
class-member-declaration
class-member-declarations class-member-declaration

class-member-declaration:
constant-declaration (not implemented)
field-declaration
method-declaration
property-declaration
event-declaration (not implemented)
indexer-declaration (not implemented)
operator-declaration (not implemented)
constructor-declaration
destructor-declaration
static-constructor-declaration
type-declaration

field-declaration:
attributes(optional)field-modifiers(optional)type (optional) variable-declarators ;

field-modifiers:
field-modifier
field-modifiers field-modifier

field-modifier:
new
public
protected
internal
private
static
readonly

variable-declarators:
variable-declarator
variable-declarators , variable-declarator

variable-declarator:
identifier
identifier = variable-initializer

variable-initializer:
    expression
    array-initializer

method-declaration:
    method-header method-body

method-header:
    attributes(optional)method-modifiers(optional) keyword(optional) return-type(optional) return-identifier = (optional) member-name (formal-parameter-list(optional))

method-modifiers:
    method-modifier
    method-modifiers method-modifier

method-modifier:
    new
    public
    protected
    internal
    private
    definition
    static
    virtual
    sealed
    override
    abstract
    extern

keyword:
    function
    ƒ

return-type:
    type
    void

member-name:
    identifier
    interface-type . identifier

method-body:
    block
    ;
    ◇

formal-parameter-list:
    fixed-parameters
    fixed-parameters , parameter-array
    parameter-array
    default-parameters

fixed-parameters:
    fixed-parameter
    fixed-parameters , fixed-parameter

fixed-parameter:
    attributes(optional)parameter-modifier(optional)type identifier

parameter-modifier:
    ref
    out

default-parameters:
    identifer = expression
    fixed parameters, default-parameters

parameter-array:
    attributes(optional)params array-type identifier

property-declaration:
    attributes(optional)property-modifiers(optional) keyword(optional) type(optional) member-name {

accessor-declarations }

property-modifiers:
property-modifier
property-modifiers property-modifier

property-modifier:
new
public
protected
internal
private
static
virtual
sealed
override
abstract
extern

member-name:
identifier
interface-type . identifier

accessor-declarations:
get-accessor-declaration set-accessor-declarationopt
set-accessor-declaration get-accessor-declaration(optional)

get-accessor-declaration:
attributes(optional)get accessor-body
set-accessor-declaration:
attributes(optional)set accessor-body

accessor-body:
block
;
◇

constructor-declaration:
attributes(optional)constructor-modifiers(optional)constructor-declarator constructor-body

constructor-modifiers:
constructor-modifier
constructor-modifiers constructor-modifier

constructor-modifier:
public
protected
internal
private
extern

constructor-declarator:
identifier (formal-parameter-list(optional)) constructor-initializer(optional)

constructor-initializer: (not implemented)
: base *(argument-list(optional))*
: this *(argument-list(optional))*

constructor-body:
block
;
◇

static-constructor-declaration:
attributes(optional)static-constructor-modifiers identifier ( ) static-constructor-body
static-constructor-modifiers
extern*(optional)*static
static extern*(optional)*

static-constructor-body:
block
;

◇

destructor-declaration:
attributes(optional)extern(optional)~ identifier ( ) destructor-body

destructor-body:
block
;
◇

array-type:
    non-array-type rank-specifiers

non-array-type:
    type (optional)

rank-specifiers:
    rank-specifier (optional)
    rank-specifiers rank-specifier

rank-specifier:
    [*dim-separators(optional)*]

dim-separators:
    , ;
    dim-separators , ;

array-initializer:
    { *variable-initializer-list(optional)*}
    { *variable-initializer-list , ;* }

variable-initializer-list:
    variable-initializer
    variable-initializer-list , variable-initializer
    variable-initializer-list ; variable-initializer

variable-initializer:
    expression
    array-initializer

A.2.9 Interfaces

interface-declaration:
attributes(optional)interface-modifiers(optional)interface identifier
interface-base(optional)interface-body ;(optional)

interface-modifiers:
interface-modifier
interface-modifiers interface-modifier

interface-modifier:
new
public
protected
internal
private
interface-base :
: *interface-type-list*

interface-body:
{ *interface-member-declarations(optional)*}

interface-member-declarations:
interface-member-declaration
interface-member-declarations interface-member-declaration

interface-member-declaration:
interface-method-declaration
interface-property-declaration
interface-event-declaration
interface-indexer-declaration

interface-method-declaration:
attributes(optional)new(optional)return-type identifier (formal-parameter-list(optional)) ;

interface-property-declaration:
attributes(optional)new(optional)type identifier { interface-accessors }

interface-accessors:
attributes(optional) get ;
attributes(optional) set ;
attributes(optional) get ; *attributes(optional)* set ;
attributes(optional) set ; *attributes(optional)* get ;

interface-event-declaration:
attributes(optional)new(optional)event type identifier ;

interface-indexer-declaration:
attributes(optional) new *(optional) type* this [*formal-parameter-list*] { *interface-accessors* }

In this section you will find a precise overview of the syntactic grammar of the Visual APL programming language.

token:
symbol
keyword
identifier-or-name
constants-or-literals
operator-or-punctuactor

new-line:
Carriage return character (U+000D)
Line feed character (U+000A)
Carriage return character (U+000D) followed by line feed character (U+000A)

Abstract

This document describes the syntax, semantics, and design of the Visual APL programming language.

Navigation

To navigate within this document, you may either use the document tree to the left, or view the summary page which contains the same information flattened to a single page.

This document contains all summaries of the lexical and syntactic grammars found in the document tree to the left. Grammar productions appear here in the same order that they appear in the navigation tree.

A.1 Lexical grammar

input:
input-section (optional)

input-section:
input-section-part
input-section input-section-part

input-section-part:
input-elements (optional)new-line
pp-directive

input-elements:
input-element
input-elements input-element

input-element:
whitespace
comment
token

A.1.1 Line terminators

new-line:
Carriage return character (U+000D)
Line feed character (U+000A)
Carriage return character (U+000D) followed by line feed character (U+000A)

A.1.2 White space

whitespace:
Any character with Unicode class Zs
Horizontal tab character (U+0009)
Vertical tab character (U+000B)
Form feed character (U+000C)

A.1.3 Comments

comment:
single-line-comment
delimited-comment

single-line-comment:
// input-characters(optional)
input-characters(optional)

input-characters:
input-character
input-characters input-character

input-character:
Any Unicode character except a new-line-character

new-line-character:
Carriage return character (U+000D)
Line feed character (U+000A)

delimited-comment:
/ delimited-comment-characters (optional) /

delimited-comment-characters:
delimited-comment-character
delimited-comment-characters delimited-comment-character

delimited-comment-character:
not-lamp
not-slash

not-lamp:
Any Unicode character except

not-slash:
Any Unicode character except /

A.1.4 Tokens

token:
symbol
keyword
identifier-or-name
constants-or-literals
operator-or-punctuactor

A.1.5 Unicode character escape sequences

unicode-escape-sequence:
\u hex-digit hex-digit hex-digit hex-digit
\U hex-digit hex-digit hex-digit hex-digit hex-digit hex-digit hex-digit hex-digit

A.1.6 Identifiers

identifier:
available-identifier
@ identifier-or-keyword - not implemented (planned)

available-identifier:
An identifier-or-keyword that is not a keyword

identifier-or-keyword:
identifier-start-character identifier-part-characters(optional)

identifier-start-character:
letter-character
_ (the underscore character U+005F)

identifier-part-characters:
identifier-part-character
identifier-part-characters identifier-part-character

identifier-part-character:
letter-character
decimal-digit-character
connecting-character
combining-character
formatting-character

letter-character:
A Unicode character of classes Lu, Ll, Lt, Lm, Lo, or Nl
A unicode-escape-sequence representing a character of classes Lu, Ll, Lt, Lm, Lo, or Nl

combining-character:
A Unicode character of classes Mn or Mc
A unicode-escape-sequence representing a character of classes Mn or Mc

decimal-digit-character:
A Unicode character of the class Nd
A unicode-escape-sequence representing a character of the class Nd

connecting-character:
A Unicode character of the class Pc
A unicode-escape-sequence representing a character of the class Pc

formatting-character:
A Unicode character of the class Cf
A unicode-escape-sequence representing a character of the class Cf

A.1.7 Keywords

keyword: one of

abstract	as	base	bool	break	byte	case
catch	char	checked	class	const	continue	decimal
default	delegate	do	double	else	enum	event
explicit	extern	false	finally	fixed	float	for
foreach	goto	if	implicit	in	int	interface
internal	is	lock	long	namespace	new	null
object	operator	out	override	params	private	protected
public	readonly	ref	return	sbyte	sealed	short
sizeof	stackalloc	static	string	struct	switch	this
throw	true	try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	void	volatile	while
and	function	elseif	then	eval	global	step
not	or	nop	print	yield	repeat	until
definition	classtype	classop	optype	_arglist	_argnames	
referencebyname	referencebyfile					

A.1.8 Literals

literal:
boolean-literal
integer-literal
real-literal
character-literal
string-literal
null-literal
type-literal

boolean-literal:
true
false

integer-literal:
decimal-integer-literal
hexadecimal-integer-literal

decimal-integer-literal:
decimal-digits integer-type-suffix(optional)

decimal-digits:
decimal-digit
decimal-digits decimal-digit

decimal-digit: one of
0 1 2 3 4 5 6 7 8 9

integer-type-suffix: one of
U u L l UL Ul uL ul LU Lu lU lu

hexadecimal-integer-literal:
    0x hex-digits integer-type-suffix(optional)
    0X hex-digits integer-type-suffix(optional)

hex-digits:
    hex-digit
    hex-digits hex-digit

hex-digit: one of
    0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f

real-literal:
    decimal-digits . decimal-digits exponent-part(optional)real-type-suffix (optional)
    . decimal-digits exponent-part(optional)real-type-suffix (optional)
    decimal-digits exponent-part real-type-suffix (optional)
    decimal-digits real-type-suffix

exponent-part:
    e sign(optional)decimal-digits
    E sign(optional)decimal-digits

sign: one of
    + -¯

real-type-suffix: one of
    F f D d M m

character-literal:
    'c'
    ' character '

character:
    single-character
    simple-escape-sequence
    hexadecimal-escape-sequence
    unicode-escape-sequence

single-character:
    Any character except ' (U+0027), \ (U+005C), and new-line-character

simple-escape-sequence: one of
    \' \" \\ \0 \a \b \f \n \r \t \v

hexadecimal-escape-sequence:
    \x hex-digit hex-digit(optional)hex-digit(optional)hex-digit(optional)

characters:
    array-characters
    array-simple-escape-sequences
    array-hexadecimal-escape-sequences
    array-unicode-escape-sequences

array-character:
    Any characters except ' (U+0027), \ (U+005C), and new-line-character

array-simple-escape-sequences: any of
    \' \" \\ \0 \a \b \f \n \r \t \v

array-hexadecimal-escape-sequence:
    \x hex-digit hex-digit(optional)hex-digit(optional)hex-digit(optional)

verbatim-characters:
    @' verbatim -array-literal-characters(optional)'

    verbatim-array-literal-characters:
    verbatim-array-literal-character
    verbatim-array-literal-characters
    verbatim-array-literal-character

verbatim-array-literal-character:
single-verbatim-array-literal-character
quote-escape-sequence

single-verbatim-array-literal-character:
any character except '

string-literal:
regular-string-literal
verbatim-string-literal

regular-string-literal:
" regular-string-literal-characters(optional)"

regular-string-literal-characters:
regular-string-literal-character
regular-string-literal-characters
regular-string-literal-character

regular-string-literal-character:
single-regular-string-literal-character
simple-escape-sequence
hexadecimal-escape-sequence
unicode-escape-sequence

single-regular-string-literal-character:
Any character except " (U+0022), \ (U+005C), and new-line-character

verbatim-string-literal:
@" verbatim -string-literal-characters(optional)"

verbatim-string-literal-characters:
verbatim-string-literal-character
verbatim-string-literal-characters
verbatim-string-literal-character

verbatim-string-literal-character:
single-verbatim-string-literal-character
quote-escape-sequence

single-verbatim-string-literal-character:
any character except "

quote-escape-sequence:
""

null-literal:
null

A.1.9 Operators and punctuators

operator-or-punctuator: one of
{ } [ ] ( ) . , : ;
+ - * / % & | ^ ! ~
= < > ? ++ -- && || << >>
== != <= >= += -= *= /= %= &=
|= ^= <<= >>= ->
◇ ¨ ¯ ≤ ≈ ≥ ≠ ∨ ∧ × ÷ ϵ ↑ ↓ ○ * ← →
⌈ ⌊ ƒ ∇ Δ ∘ ≡

A.1.10 Pre-processing directives

pp-directive:
pp-declaration (not implemented)
pp-conditional (not implemented)
pp-line (not implemented)
pp-diagnostic (not implemented)

pp-region

pp-new-line:
whitespace(optional)single-line-comment(optional)new-line

skipped-characters:
whitespace(optional)not-number-sign input-characters(optional)

not-number-sign:
Any input-character except #

pp-region:
pp-start-region conditional-section(optional)pp-end-region

pp-start-region:
whitespace(optional)# whitespace(optional)region pp-message

pp-end-region:
whitespace(optional)# whitespace(optional)endregion pp-message

A.2 Syntactic grammar

A.2.1 Basic concepts

namespace-name:
namespace-or-type-name

type-name:
namespace-or-type-name

namespace-or-type-name:
identifier
namespace-or-type-name . identifier

A.2.2 Types

type:
value-type
reference-type

value-type:
enum-type

struct-type:
type-name
simple-type

simple-type:
numeric-type
bool

numeric-type:
integral-type
floating-point-type
decimal
integral-type:
sbyte
byte
short
ushort
int
uint
long
ulong
char

floating-point-type:

float
double

enum-type:
type-name

reference-type:
class-type
interface-type
array-type
delegate-type

class-type:
type-name
object
string

interface-type:
type-name

array-type:
non-array-type rank-specifiers

non-array-type:
type

rank-specifiers:
rank-specifier
rank-specifiers rank-specifier

rank-specifier:
[dim-separators(optional)]

dim-separators:
,
dim-separators ,
;
dim-separators ;

delegate-type:
type-name

A.2.3 Variables

variable-reference:
expression

A.2.4 Expressions

argument-list:
argument
argument-list , argument

argument:
expression
ref variable-reference
out variable-reference

primary-expression:
primary-no-array-creation-expression
array-creation-expression

primary-no-array-creation-expression:
literal
simple-name
parenthesized-expression
member-access
invocation-expression

element-access
this-access
base-access
post-increment-expression
post-decrement-expression
object-creation-expression
delegate-creation-expression
typeof-expression
checked-expression
unchecked-expression

simple-name:
identifier

parenthesized-expression:
(expression)

member-access:
primary-expression . identifier
predefined-type . identifier
predefined-type: one of
bool byte char decimal double float int long
object sbyte short string uint ulong ushort

invocation-expression:
primary-expression (argument-list(optional))

element-access:
primary-no-array-creation-expression [expression-list]

expression-list:
expression
expression-list , expression

this-access:
this

base-access:
base . *identifier*
base [*expression-list*]

post-increment-expression:
primary-expression ++

post-decrement-expression:
primary-expression --

object-creation-expression:
new type (argument-list(optional))
type (argument-list(optional))

array-creation-expression:
new *non-array-type [expression-list] rank-specifiers(optional)array-initializer (optional)*
new *array-type array-initializer*

delegate-creation-expression:
expression

typeof-expression:
typeof (*type*)
typeof (void)

checked-expression:
checked (*expression*)
unchecked-expression:
unchecked (*expression*)

unary-expression:
primary-expression
monadic-expressions
+ monadic-expression
- monadic-expression

! monadic-expression
~ monadic-expression
** monadic-expression*
≤monadic-expression
≈monadic-expression
≥monadic-expression
≠monadic-expression
⋁ monadic-expression
⋀ monadic-expression
× monadic-expression
÷ monadic-expression
∈ monadic-expression
monadic-expression
¯ monadic-expression
↑ monadic-expression
↓ monadic-expression
monadic-expression
○ monadic-expression
⌈ monadic-expression
⌊ monadic-expression
monadic-expression
monadic-expression
⊂ monadic-expression
⊃ monadic-expression
∣ monadic-expression
monadic-expression
≡ monadic-expression
monadic-expression
monadic-expression
monadic-expression
monadic-expression
⊖ monadic-expression
monadic-expression
monadic-expression
monadic-expression
monadic-expression
cast-expression

cast-expression:
(type) unary-expression

multiplicative-expression:
unary-expression
multiplicative-expression × unary-expression
multiplicative-expression ÷ unary-expression
multiplicative-expression % unary-expression
multiplicative-expression ∣ unary-expression

additive-expression:
multiplicative-expression
additive-expression + multiplicative-expression
additive-expression multiplicative-expression

shift-expression:
additive-expression
shift-expression << additive-expression
shift-expression >> additive-expression

relational-expression:
shift-expression
relational-expression < shift-expression
relational-expression > shift-expression
relational-expression <= shift-expression
relational-expression ≤shift-expression
relational-expression >= shift-expression
relational-expression ≤shift-expression
relational-expression is type
relational-expression as type

equality-expression:
relational-expression

equality-expression == relational-expression
equality-expression ≈relational-expression
equality-expression ≡ relational-expression
equality-expression != relational-expression
equality-expression ≠relational-expression

and-expression:
equality-expression
and-expression & equality-expression
and-expression ∧ equality-expression

inclusive-or-expression:
inclusive-or-expression | exclusive-or-expression

conditional-and-expression:
inclusive-or-expression
conditional-and-expression && inclusive-or-expression

conditional-or-expression:
conditional-and-expression
conditional-or-expression || conditional-and-expression

conditional-expression:
conditional-or-expression
conditional-or-expression then expression else expression

array-scalar-expression:
array-scalar-expression + monadic-expression
array-scalar-expression - monadic-expression
array-scalar-expression ! monadic-expression
array-scalar-expression ~ monadic-expression
*array-scalar-expression * monadic-expression*
array-scalar-expression ≤monadic-expression
array-scalar-expression ≈monadic-expression
array-scalar-expression ≥monadic-expression
array-scalar-expression ≠monadic-expression
array-scalar-expression ∨ monadic-expression
array-scalar-expression ∧ monadic-expression
array-scalar-expression × monadic-expression
array-scalar-expression ÷ monadic-expression
array-scalar-expression ϵ monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression ~ monadic-expression
array-scalar-expression ↑ monadic-expression
array-scalar-expression ↓ monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression ○ monadic-expression
array-scalar-expression ⌈ monadic-expression
array-scalar-expression ⌊ monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression ⊂ monadic-expression
array-scalar-expression ⊃ monadic-expression
array-scalar-expression | monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression ≡ monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression ⊖ monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression monadic-expression
array-scalar-expression monadic-expression

assignment:
unary-expression assignment-operator expression

assignment-operator: one of
= += -= *= %= &= |= ^= <<= >>=

≤ ≈ ≥ ≠ ⋁= ⋀= ×= ÷= ϵ= = ~=
↑= ↓= = ○=
⌈= ⌊= ⊂= ⊃= ⊥= ⊤= |= =
≡= = =
= = ⊖= = = = =

expression:
conditional-expression
assignment

constant-expression:
expression

boolean-expression:
expression

A.2.5 Statements

statement:
labeled-statement
declaration-statement
embedded-statement

embedded-statement:
block
empty-statement
expression-statement
selection-statement
iteration-statement
jump-statement
try-statement
checked-statement
unchecked-statement
lock-statement (not implemented)

block:
{ statement-list(optional)}

statement-list:
statement
statement-list statement

empty-statement:
; (optional)
◇ (optional)

labeled-statement:
identifier : statement

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

local-variable-declaration:
type local-variable-declarators

local-variable-declarators:
local-variable-declarator
local-variable-declarators , local-variable-declarator

local-variable-declarator:
identifier (optional)
identifier = local-variable-initializer

local-variable-initializer:
expression
array-initializer

local-constant-declaration:

const *type constant-declarators*

constant-declarators:
constant-declarator
constant-declarators , constant-declarator

constant-declarator:
identifier = constant-expression

expression-statement:
statement-expression ;

statement-expression:
invocation-expression
object-creation-expression
assignment
post-increment-expression
post-decrement-expression
pre-increment-expression
pre-decrement-expression

selection-statement:
if-statement
switch-statement

if-statement:
if *(boolean-expression*) *embedded-statement*
if (*boolean-expression*) *embedded-statement else embedded-statement*

boolean-expression:
expression

switch-statement:
switch (*expression*) *switch-block*

switch-block:
{ *switch-sections(optional)}*

switch-sections:
switch-section
switch-sections switch-section

switch-section:
switch-labels statement-list

switch-labels:
switch-label
switch-labels switch-label

switch-label:
case *expression* :
default :

iteration-statement:
while-statement
do-statement
for-statement
foreach-statement

while-statement:
while (*boolean-expression*) *embedded-statement*

do-statement:
do *embedded-statement* while (*boolean-expression*) ; *(optional)*

for-statement:
for (*for-initializer(optional); for-condition(optional*; *for-iterator(optional)) embedded-statement*

for-initializer:
local-variable-declaration
statement-expression-list

for-condition:
    boolean-expression

for-iterator:
    statement-expression-list

statement-expression-list:
    statement-expression
    statement-expression-list , statement-expression

foreach-statement:
    foreach (*type(optional) identifier in expression*) *embedded-statement*

jump-statement:
    break-statement
    continue-statement
    goto-statement
    return-statement
    throw-statement

break-statement:
    break

continue-statement:
    continue

goto-statement:
    goto *identifier*

return-statement:
    return *expression(optional)*;

throw-statement:
    throw *expression(optional)*

try-statement:
    try *block catch-clauses*
    try *block finally-clause*
    try *block catch-clauses finally-clause*

catch-clauses:
    specific-catch-clauses general-catch-clauseopt
    specific-catch-clauses(optional)general-catch-clause

specific-catch-clauses:
    specific-catch-clause
    specific-catch-clauses specific-catch-clause

specific-catch-clause:
    catch (*class-type identifier(optional)*) *block*

general-catch-clause:
    catch *block*

finally-clause:
    finally *block*

checked-statement:
    checked *block*

unchecked-statement:
    unchecked *block*

lock-statement: (not implemented)
    lock (*expression*) *embedded-statement*

resource-acquisition:
    local-variable-declaration
    expression

A.2.6 Namespaces

compilation-unit:
    using-directives(optional)global-attributes(optional)namespace-member-declarations(optional)
    reference-directives(optional)namespace-member-declarations(optional)

namespace-declaration:
    namespace *qualified-identifier namespace-body* ;(optional)

qualified-identifier:
    identifier
    qualified-identifier . identifier

namespace-body:
    { *reference-directives(optional)namespace-member-declarations(optional)*}
    { *using-directives(optional)namespace-member-declarations(optional)*}

reference-directives:
    reference-directive
    referencebyfile filename
    referencebyfile using-directive

using-directives:
    using-directive
    using-directives using-directive

using-directive:
    using-alias-directive
    using-namespace-directive

using-alias-directive:
    using *identifier = namespace-or-type-name* ;
    using *namespace-or-type-name* as *identifier (depricated)*

using-namespace-directive:
    using *namespace-name* ;

namespace-member-declarations:
    namespace-member-declaration
    namespace-member-declarations namespace-member-declaration

namespace-member-declaration:
    namespace-declaration
    type-declaration

type-declaration:
    class-declaration
    interface-declaration
    enum-declaration

A.2.7 Classes

class-declaration:
    attributes(optional)class-modifiers(optional)class identifier class-base(optional)class-body ;(optional)

class-modifiers:
    class-modifier
    class-modifiers class-modifier

class-modifier:
    new
    public
    protected
    internal
    private
    abstract
    sealed

class-base:
: *class-type*
: *interface-type-list*
: *class-type , interface-type-list*

interface-type-list:
interface-type
interface-type-list , interface-type

class-body:
{ *class-member-declarations(optional)*}

class-member-declarations:
class-member-declaration
class-member-declarations class-member-declaration

class-member-declaration:
constant-declaration (not implemented)
field-declaration
method-declaration
property-declaration
event-declaration (not implemented)
indexer-declaration (not implemented)
operator-declaration (not implemented)
constructor-declaration
destructor-declaration
static-constructor-declaration
type-declaration

field-declaration:
attributes(optional)field-modifiers(optional)type (optional) variable-declarators ;

field-modifiers:
field-modifier
field-modifiers field-modifier

field-modifier:
new
public
protected
internal
private
static
readonly

variable-declarators:
variable-declarator
variable-declarators , variable-declarator

variable-declarator:
identifier
identifier = variable-initializer

variable-initializer:
expression
array-initializer

method-declaration:
method-header method-body

method-header:
attributes(optional)method-modifiers(optional) keyword(optional) return-type(optional) return-identifier = (optional) member-name (formal-parameter-list(optional))

method-modifiers:
method-modifier
method-modifiers method-modifier

method-modifier:
new
public
protected

internal
private
definition
static
virtual
sealed
override
abstract
extern

keyword:
function
ƒ

return-type:
type
void

member-name:
identifier
interface-type . identifier

method-body:
block
;
◇

formal-parameter-list:
fixed-parameters
fixed-parameters , parameter-array
parameter-array
default-parameters

fixed-parameters:
fixed-parameter
fixed-parameters , fixed-parameter

fixed-parameter:
attributes(optional)parameter-modifier(optional)type identifier

parameter-modifier:
ref
out

default-parameters:
identifer = expression
fixed parameters, default-parameters

parameter-array:
attributes(optional)params array-type identifier

property-declaration:
attributes(optional)property-modifiers(optional) keyword(optional) type(optional) member-name {
accessor-declarations }

property-modifiers:
property-modifier
property-modifiers property-modifier

property-modifier:
new
public
protected
internal
private
static
virtual
sealed
override
abstract
extern

member-name:
    identifier
    interface-type . identifier

accessor-declarations:
    get-accessor-declaration set-accessor-declarationopt
    set-accessor-declaration get-accessor-declaration(optional)

get-accessor-declaration:
    attributes(optional)get accessor-body
    set-accessor-declaration:
    attributes(optional)set accessor-body

accessor-body:
    block
    ;
    ◇

constructor-declaration:
    attributes(optional)constructor-modifiers(optional)constructor-declarator constructor-body

constructor-modifiers:
    constructor-modifier
    constructor-modifiers constructor-modifier

constructor-modifier:
    public
    protected
    internal
    private
    extern

constructor-declarator:
    identifier (formal-parameter-list(optional)) constructor-initializer(optional)

constructor-initializer: (not implemented)
    : base *(argument-list(optional))*
    : this *(argument-list(optional))*

constructor-body:
    block
    ;
    ◇

static-constructor-declaration:
    attributes(optional)static-constructor-modifiers identifier ( ) static-constructor-body
    static-constructor-modifiers
    extern*(optional)*static
    static extern*(optional)*

static-constructor-body:
    block
    ;
    ◇

destructor-declaration:
    attributes(optional)extern(optional)~ identifier ( ) destructor-body

destructor-body:
    block
    ;
    ◇

A.2.8 Arrays

array-type:
    non-array-type rank-specifiers

non-array-type:
    type (optional)

rank-specifiers:
rank-specifier (optional)
rank-specifiers rank-specifier

rank-specifier:
[*dim-separators(optional)*]

dim-separators:
, ;
dim-separators , ;

array-initializer:
{ *variable-initializer-list(optional)*}
{ *variable-initializer-list , ;* }

variable-initializer-list:
variable-initializer
variable-initializer-list , variable-initializer
variable-initializer-list ; variable-initializer

variable-initializer:
expression
array-initializer

A.2.9 Interfaces

interface-declaration:
attributes(optional)interface-modifiers(optional)interface identifier
interface-base(optional)interface-body ;(optional)

interface-modifiers:
interface-modifier
interface-modifiers interface-modifier

interface-modifier:
new
public
protected
internal
private
interface-base :
: *interface-type-list*

interface-body:
{ *interface-member-declarations(optional)*}

interface-member-declarations:
interface-member-declaration
interface-member-declarations interface-member-declaration

interface-member-declaration:
interface-method-declaration
interface-property-declaration
interface-event-declaration
interface-indexer-declaration

interface-method-declaration:
attributes(optional)new(optional)return-type identifier (formal-parameter-list(optional)) ;

interface-property-declaration:
attributes(optional)new(optional)type identifier { interface-accessors }

interface-accessors:
attributes(optional) get ;
attributes(optional) set ;
attributes(optional) get ; *attributes(optional)* set ;
attributes(optional) set ; *attributes(optional)* get ;

interface-event-declaration:

attributes(optional)new(optional)event type identifier ;

interface-indexer-declaration:
attributes(optional) new *(optional) type* this [*formal-parameter-list*] { *interface-accessors* }

A.2.10 Enums

enum-declaration:
attributes(optional)enum-modifiers(optional)enum identifier enum-base(optional)enum-body ;(optional)

enum-base:
: *integral-type*

enum-body:
{ *enum-member-declarations(optional)*}
{ *enum-member-declarations ,* }

enum-modifiers:
enum-modifier
enum-modifiers enum-modifier

enum-modifier:
new
public
protected
internal
private

enum-member-declarations:
enum-member-declaration
enum-member-declarations , enum-member-declaration

enum-member-declaration:
attributes(optional)identifier
attributes(optional)identifier = constant-expression

A.2.11 Attributes

global-attributes:
global-attribute-sections

global-attribute-sections:
global-attribute-section
global-attribute-sections global-attribute-section

global-attribute-section:
[*global-attribute-target-specifier attribute-list*]
[*global-attribute-target-specifier attribute-list ,*]

global-attribute-target-specifier:
global-attribute-target :
global-attribute-target:
assembly
module

attributes:
attribute-sections

attribute-sections:
attribute-section
attribute-sections attribute-section

attribute-section:
[*attribute-target-specifier(optional)attribute-list*]
[*attribute-target-specifier(optional)attribute-list ,*]

attribute-target-specifier:
attribute-target :

attribute-target:

field
event (not implemented)
method
param (not implemented)
property
return (not implemented)
type

attribute-list:
attribute
attribute-list , attribute

attribute:
attribute-name attribute-arguments(optional)

attribute-name:
type-name

attribute-arguments:
(positional-argument-list(optional))
(positional-argument-list , named-argument-list)
(named-argument-list)

positional-argument-list:
positional-argument
positional-argument-list , positional-argument

positional-argument:
attribute-argument-expression

named-argument-list:
named-argument
named-argument-list , named-argument

named-argument:
identifier = attribute-argument-expression
attribute-argument-expression:
expression