# C. Grammar

**Visual Studio .NET 2003**

This appendix contains summaries of the lexical and syntactic grammars found in the main document, and of the grammar extensions for unsafe code. Grammar productions appear here in the same order that they appear in the main document.

## C.1 Lexical grammar

*input:*
    *input-section*$_{opt}$
*input-section:*
    *input-section-part*
    *input-section* *input-section-part*
*input-section-part:*
    *input-elements*$_{opt}$ *new-line*
    *pp-directive*
*input-elements:*
    *input-element*
    *input-elements* *input-element*
*input-element:*
    *whitespace*
    *comment*
    *token*

## C.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)
    Line separator character (U+2028)
    Paragraph separator character (U+2029)

## C.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)

## C.1.3 Comments

*comment:*
    *single-line-comment*
    *delimited-comment*
*single-line-comment:*

*// input-characters$_{opt}$*

*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)

Line separator character (U+2028)

Paragraph separator character (U+2029)

*delimited-comment:*

*/* delimited-comment-characters$_{opt}$ */*

*delimited-comment-characters:*

*delimited-comment-character*

*delimited-comment-characters delimited-comment-character*

*delimited-comment-character:*

*not-asterisk*

** not-slash*

*not-asterisk:*

Any Unicode character except *

*not-slash:*

Any Unicode character except /

# C.1.4 Tokens

*token:*

*identifier*

*keyword*

*integer-literal*

*real-literal*

*character-literal*

*string-literal*

*operator-or-punctuator*

# C.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*

# C.1.6 Identifiers

*identifier:*

*available-identifier*

*@ identifier-or-keyword*

*available-identifier:*

An *identifier-or-keyword* that is not a *keyword*

*identifier-or-keyword:*

*identifier-start-character identifier-part-characters$_{opt}$*

*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

# C.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 |

# C.1.8 Literals

*literal:*
    *boolean-literal*
    *integer-literal*
    *real-literal*
    *character-literal*
    *string-literal*
    *null-literal*
*boolean-literal:*
    true
    false
*integer-literal:*
    *decimal-integer-literal*
    *hexadecimal-integer-literal*
*decimal-integer-literal:*
    *decimal-digits* *integer-type-suffix*$_{opt}$
*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*$_{opt}$
    0X *hex-digits* *integer-type-suffix*$_{opt}$
*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*$_{opt}$ *real-type-suffix*$_{opt}$
    . *decimal-digits* *exponent-part*$_{opt}$ *real-type-suffix*$_{opt}$
    *decimal-digits* *exponent-part* *real-type-suffix*$_{opt}$
    *decimal-digits* *real-type-suffix*
*exponent-part:*
    e *sign*$_{opt}$ *decimal-digits*
    E *sign*$_{opt}$ *decimal-digits*
*sign:* one of
    + -
*real-type-suffix:* one of
    F f D d M m
*character-literal:*
    ' *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*$_{opt}$ *hex-digit*$_{opt}$ *hex-digit*$_{opt}$
*string-literal:*

*regular-string-literal*
*verbatim-string-literal*

*regular-string-literal:*
" *regular-string-literal-characters*$_{opt}$ "

*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*$_{opt}$ "

*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

# C.1.9 Operators and punctuators

*operator-or-punctuator:* one of
{ } [ ] ( ) . , : ;
+ - * / % & | ^ ! ~
= < > ? ++ -- && || << >>
== != <= >= += -= *= /= %= &=
|= ^= <<= >>= ->

# C.1.10 Pre-processing directives

*pp-directive:*
*pp-declaration*
*pp-conditional*
*pp-line*
*pp-diagnostic*
*pp-region*

*pp-new-line:*
*whitespace*$_{opt}$ *single-line-comment*$_{opt}$ *new-line*

*conditional-symbol:*
Any *identifier-or-keyword* except true or false

*pp-expression:*
*whitespace*$_{opt}$ *pp-or-expression* *whitespace*$_{opt}$

*pp-or-expression:*
*pp-and-expression*
*pp-or-expression* *whitespace*$_{opt}$ || *whitespace*$_{opt}$ *pp-and-expression*

*pp-and-expression:*
*pp-equality-expression*
*pp-and-expression* *whitespace*$_{opt}$ && *whitespace*$_{opt}$ *pp-equality-expression*
*pp-equality-expression:*
*pp-unary-expression*
*pp-equality-expression* *whitespace*$_{opt}$ == *whitespace*$_{opt}$ *pp-unary-expression*
*pp-equality-expression* *whitespace*$_{opt}$ != *whitespace*$_{opt}$ *pp-unary-expression*
*pp-unary-expression:*
*pp-primary-expression*
! *whitespace*$_{opt}$ *pp-unary-expression*
*pp-primary-expression:*
true
false
*conditional-symbol*
( *whitespace*$_{opt}$ *pp-expression* *whitespace*$_{opt}$ )
*pp-declaration:*
*whitespace*$_{opt}$ # *whitespace*$_{opt}$ define *whitespace* *conditional-symbol* *pp-new-line*
*whitespace*$_{opt}$ # *whitespace*$_{opt}$ undef *whitespace* *conditional-symbol* *pp-new-line*
*pp-conditional:*
*pp-if-section* *pp-elif-sections*$_{opt}$ *pp-else-section*$_{opt}$ *pp-endif*
*pp-if-section:*
*whitespace*$_{opt}$ # *whitespace*$_{opt}$ if *whitespace* *pp-expression* *pp-new-line* *conditional-section*$_{opt}$
*pp-elif-sections:*
*pp-elif-section*
*pp-elif-sections* *pp-elif-section*
*pp-elif-section:*
*whitespace*$_{opt}$ # *whitespace*$_{opt}$ elif *whitespace* *pp-expression* *pp-new-line* *conditional-section*$_{opt}$
*pp-else-section:*
*whitespace*$_{opt}$ # *whitespace*$_{opt}$ else *pp-new-line* *conditional-section*$_{opt}$
*pp-endif:*
*whitespace*$_{opt}$ # *whitespace*$_{opt}$ endif *pp-new-line*
*conditional-section:*
*input-section*
*skipped-section*
*skipped-section:*
*skipped-section-part*
*skipped-section* *skipped-section-part*
*skipped-section-part:*
*skipped-characters*$_{opt}$ *new-line*
*pp-directive*
*skipped-characters:*
*whitespace*$_{opt}$ *not-number-sign* *input-characters*$_{opt}$
*not-number-sign:*
Any *input-character* except #
*pp-line:*
*whitespace*$_{opt}$ # *whitespace*$_{opt}$ line *whitespace* *line-indicator* *pp-new-line*
*line-indicator:*
*decimal-digits* *whitespace* *file-name*
*decimal-digits*
default
*file-name:*
" *file-name-characters* "
*file-name-characters:*
*file-name-character*
*file-name-characters* *file-name-character*

*file-name-character:*
    Any *input-character* except "
*pp-diagnostic:*
    *whitespace*$_{opt}$ # *whitespace*$_{opt}$ error *pp-message*
    *whitespace*$_{opt}$ # *whitespace*$_{opt}$ warning *pp-message*
*pp-message:*
    *new-line*
    *whitespace* *input-characters*$_{opt}$ *new-line*
*pp-region:*
    *pp-start-region* *conditional-section*$_{opt}$ *pp-end-region*
*pp-start-region:*
    *whitespace*$_{opt}$ # *whitespace*$_{opt}$ region *pp-message*
*pp-end-region:*
    *whitespace*$_{opt}$ # *whitespace*$_{opt}$ endregion *pp-message*

# C.2 Syntactic grammar

## C.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*

## C.2.2 Types

*type:*
    *value-type*
    *reference-type*
*value-type:*
    *struct-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*$_{opt}$ ]
*dim-separators:*
,
*dim-separators* ,
*delegate-type:*
*type-name*

# C.2.3 Variables

*variable-reference:*
*expression*

# C.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*
*sizeof-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*$_{opt}$ )

*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*$_{opt}$ )

*array-creation-expression:*
new *non-array-type* [ *expression-list* ] *rank-specifiers*$_{opt}$ *array-initializer*$_{opt}$
new *array-type* *array-initializer*

*delegate-creation-expression:*
new *delegate-type* ( *expression* )

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

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

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

*unary-expression:*
*primary-expression*
+ *unary-expression*
- *unary-expression*
! *unary-expression*
~ *unary-expression*

** unary-expression*
*pre-increment-expression*
*pre-decrement-expression*
*cast-expression*

*pre-increment-expression:*
++ *unary-expression*

*pre-decrement-expression:*
-- *unary-expression*

*cast-expression:*
( *type* ) *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* is *type*
*relational-expression* as *type*

*equality-expression:*
*relational-expression*
*equality-expression* == *relational-expression*
*equality-expression* != *relational-expression*

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

*exclusive-or-expression:*
*and-expression*
*exclusive-or-expression* ^ *and-expression*

*inclusive-or-expression:*
*exclusive-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* ? *expression* : *expression*

*assignment:*
*unary-expression* *assignment-operator* *expression*

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

*expression:*
    *conditional-expression*
    *assignment*
*constant-expression:*
    *expression*
*boolean-expression:*
    *expression*

# C.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*
    *using-statement*
*block:*
    { *statement-list*$_{opt}$ }
*statement-list:*
    *statement*
    *statement-list* *statement*
*empty-statement:*
    ;
*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*
    *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*$_{opt}$ }
*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 *constant-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* ) ;
*for-statement:*
    for ( *for-initializer*$_{opt}$ ; *for-condition*$_{opt}$ ; *for-iterator*$_{opt}$ ) *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* *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* ;
goto case *constant-expression* ;
goto default ;

*return-statement:*
return *expression*$_{opt}$ ;

*throw-statement:*
throw *expression*$_{opt}$ ;

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

*catch-clauses:*
*specific-catch-clauses* *general-catch-clause*$_{opt}$
*specific-catch-clauses*$_{opt}$ *general-catch-clause*

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

*specific-catch-clause:*
catch ( *class-type* *identifier*$_{opt}$ ) *block*

*general-catch-clause:*
catch *block*

*finally-clause:*
finally *block*

*checked-statement:*
checked *block*

*unchecked-statement:*
unchecked *block*

*lock-statement:*
lock ( *expression* ) *embedded-statement*

*using-statement:*
using ( *resource-acquisition* ) *embedded-statement*

*resource-acquisition:*
*local-variable-declaration*
*expression*

# C.2.6 Namespaces

*compilation-unit:*
*using-directives*$_{opt}$ *global-attributes*$_{opt}$ *namespace-member-declarations*$_{opt}$

*namespace-declaration:*
namespace *qualified-identifier* *namespace-body* ;$_{opt}$

*qualified-identifier:*
*identifier*
*qualified-identifier* . *identifier*

*namespace-body:*

{ *using-directives*$_{opt}$ *namespace-member-declarations*$_{opt}$ }

*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-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*
*struct-declaration*
*interface-declaration*
*enum-declaration*
*delegate-declaration*

# C.2.7 Classes

*class-declaration:*
*attributes*$_{opt}$ *class-modifiers*$_{opt}$ class *identifier* *class-base*$_{opt}$ *class-body* ;$_{opt}$

*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*$_{opt}$ }

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

*class-member-declaration:*
*constant-declaration*
*field-declaration*
*method-declaration*
*property-declaration*

*event-declaration*
*indexer-declaration*
*operator-declaration*
*constructor-declaration*
*destructor-declaration*
*static-constructor-declaration*
*type-declaration*

*constant-declaration:*
*attributes*$_{opt}$ *constant-modifiers*$_{opt}$ const *type* *constant-declarators* ;

*constant-modifiers:*
*constant-modifier*
*constant-modifiers* *constant-modifier*

*constant-modifier:*
new
public
protected
internal
private

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

*constant-declarator:*
*identifier* = *constant-expression*

*field-declaration:*
*attributes*$_{opt}$ *field-modifiers*$_{opt}$ *type* *variable-declarators* ;

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

*field-modifier:*
new
public
protected
internal
private
static
readonly
volatile

*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*$_{opt}$ *method-modifiers*$_{opt}$ *return-type* *member-name* ( *formal-parameter-list*$_{opt}$ )

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

*method-modifier:*
new
public
protected

internal
private
static
virtual
sealed
override
abstract
extern

*return-type:*
*type*
void

*member-name:*
*identifier*
*interface-type* . *identifier*

*method-body:*
*block*
;

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

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

*fixed-parameter:*
$attributes_{opt}$ $parameter\text{-}modifier_{opt}$ *type* *identifier*

*parameter-modifier:*
ref
out

*parameter-array:*
$attributes_{opt}$ params *array-type* *identifier*

*property-declaration:*
$attributes_{opt}$ $property\text{-}modifiers_{opt}$ *type* *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* $set\text{-}accessor\text{-}declaration_{opt}$
*set-accessor-declaration* $get\text{-}accessor\text{-}declaration_{opt}$

*get-accessor-declaration:*
$attributes_{opt}$ get *accessor-body*

*set-accessor-declaration:*
    *attributes*$_{opt}$ set *accessor-body*
*accessor-body:*
    *block*
    ;
*event-declaration:*
    *attributes*$_{opt}$ *event-modifiers*$_{opt}$ event *type* *variable-declarators* ;
    *attributes*$_{opt}$ *event-modifiers*$_{opt}$ event *type* *member-name* { *event-accessor-declarations* }
*event-modifiers:*
    *event-modifier*
    *event-modifiers* *event-modifier*
*event-modifier:*
    new
    public
    protected
    internal
    private
    static
    virtual
    sealed
    override
    abstract
    extern
*event-accessor-declarations:*
    *add-accessor-declaration* *remove-accessor-declaration*
    *remove-accessor-declaration* *add-accessor-declaration*
*add-accessor-declaration:*
    *attributes*$_{opt}$ add *block*
*remove-accessor-declaration:*
    *attributes*$_{opt}$ remove *block*
*indexer-declaration:*
    *attributes*$_{opt}$ *indexer-modifiers*$_{opt}$ *indexer-declarator* { *accessor-declarations* }
*indexer-modifiers:*
    *indexer-modifier*
    *indexer-modifiers* *indexer-modifier*
*indexer-modifier:*
    new
    public
    protected
    internal
    private
    virtual
    sealed
    override
    abstract
    extern
*indexer-declarator:*
    *type* this [ *formal-parameter-list* ]
    *type* *interface-type* . this [ *formal-parameter-list* ]
*operator-declaration:*
    *attributes*$_{opt}$ *operator-modifiers* *operator-declarator* *operator-body*
*operator-modifiers:*
    *operator-modifier*
    *operator-modifiers* *operator-modifier*
*operator-modifier:*
    public

static
extern

*operator-declarator:*
*unary-operator-declarator*
*binary-operator-declarator*
*conversion-operator-declarator*

*unary-operator-declarator:*
*type* operator *overloadable-unary-operator* ( *type identifier* )

*overloadable-unary-operator:* one of
+ - ! ~ ++ -- true false

*binary-operator-declarator:*
*type* operator *overloadable-binary-operator* ( *type identifier* , *type identifier* )

*overloadable-binary-operator:* one of
+ - * / % & | ^ << >> == != > < >= <=

*conversion-operator-declarator:*
implicit operator *type* ( *type identifier* )
explicit operator *type* ( *type identifier* )

*operator-body:*
*block*
;

*constructor-declaration:*
*attributes*$_{opt}$ *constructor-modifiers*$_{opt}$ *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*$_{opt}$ ) *constructor-initializer*$_{opt}$

*constructor-initializer:*
: base ( *argument-list*$_{opt}$ )
: this ( *argument-list*$_{opt}$ )

*constructor-body:*
*block*
;

*static-constructor-declaration:*
*attributes*$_{opt}$ *static-constructor-modifiers identifier* ( ) *static-constructor-body*

*static-constructor-modifiers*
extern$_{opt}$ static
static extern$_{opt}$

*static-constructor-body:*
*block*
;

*destructor-declaration:*
*attributes*$_{opt}$ extern$_{opt}$ ~ *identifier* ( ) *destructor-body*

*destructor-body:*
*block*
;

# C.2.8 Structs

*struct-declaration:*
*attributes*$_{opt}$ *struct-modifiers*$_{opt}$ struct *identifier* *struct-interfaces*$_{opt}$ *struct-body* ;$_{opt}$
*struct-modifiers:*
*struct-modifier*
*struct-modifiers* *struct-modifier*
*struct-modifier:*
new
public
protected
internal
private
*struct-interfaces:*
: *interface-type-list*
*struct-body:*
{ *struct-member-declarations*$_{opt}$ }
*struct-member-declarations:*
*struct-member-declaration*
*struct-member-declarations* *struct-member-declaration*
*struct-member-declaration:*
*constant-declaration*
*field-declaration*
*method-declaration*
*property-declaration*
*event-declaration*
*indexer-declaration*
*operator-declaration*
*constructor-declaration*
*static-constructor-declaration*
*type-declaration*

# C.2.9 Arrays

*array-type:*
*non-array-type* *rank-specifiers*
*non-array-type:*
*type*
*rank-specifiers:*
*rank-specifier*
*rank-specifiers* *rank-specifier*
*rank-specifier:*
[ *dim-separators*$_{opt}$ ]
*dim-separators:*
,
*dim-separators* ,
*array-initializer:*
{ *variable-initializer-list*$_{opt}$ }
{ *variable-initializer-list* , }
*variable-initializer-list:*
*variable-initializer*
*variable-initializer-list* , *variable-initializer*
*variable-initializer:*
*expression*
*array-initializer*

# C.2.10 Interfaces

*interface-declaration:*
*attributes*$_{opt}$ *interface-modifiers*$_{opt}$ interface *identifier* *interface-base*$_{opt}$ *interface-body* ;$_{opt}$
*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*$_{opt}$ }
*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*$_{opt}$ new$_{opt}$ *return-type* *identifier* ( *formal-parameter-list*$_{opt}$ ) ;
*interface-property-declaration:*
*attributes*$_{opt}$ new$_{opt}$ *type* *identifier* { *interface-accessors* }
*interface-accessors:*
*attributes*$_{opt}$ get ;
*attributes*$_{opt}$ set ;
*attributes*$_{opt}$ get ; *attributes*$_{opt}$ set ;
*attributes*$_{opt}$ set ; *attributes*$_{opt}$ get ;
*interface-event-declaration:*
*attributes*$_{opt}$ new$_{opt}$ event *type* *identifier* ;
*interface-indexer-declaration:*
*attributes*$_{opt}$ new$_{opt}$ *type* this [ *formal-parameter-list* ] { *interface-accessors* }

# C.2.11 Enums

*enum-declaration:*
*attributes*$_{opt}$ *enum-modifiers*$_{opt}$ enum *identifier* *enum-base*$_{opt}$ *enum-body* ;$_{opt}$
*enum-base:*
: *integral-type*
*enum-body:*
{ *enum-member-declarations*$_{opt}$ }
{ *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*$_{opt}$ *identifier*
*attributes*$_{opt}$ *identifier* = *constant-expression*

# C.2.12 Delegates

*delegate-declaration:*
*attributes*$_{opt}$ *delegate-modifiers*$_{opt}$ delegate *return-type* *identifier* ( *formal-parameter-list*$_{opt}$ ) ;
*delegate-modifiers:*
*delegate-modifier*
*delegate-modifiers* *delegate-modifier*
*delegate-modifier:*
new
public
protected
internal
private

# C.2.13 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*$_{opt}$ *attribute-list* ]
[ *attribute-target-specifier*$_{opt}$ *attribute-list* , ]
*attribute-target-specifier:*
*attribute-target* :
*attribute-target:*
field
event
method
param
property
return
type
*attribute-list:*

*attribute*
*attribute-list* , *attribute*
*attribute:*
*attribute-name* *attribute-arguments*$_{opt}$
*attribute-name:*
*type-name*
*attribute-arguments:*
( *positional-argument-list*$_{opt}$ )
( *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*

# C.3 Grammar extensions for unsafe code

*class-modifier:*
*...*
unsafe
*struct-modifier:*
*...*
unsafe
*interface-modifier:*
*...*
unsafe
*delegate-modifier:*
*...*
unsafe
*field-modifier:*
*...*
unsafe
*method-modifier:*
*...*
unsafe
*property-modifier:*
*...*
unsafe
*event-modifier:*
*...*
unsafe
*indexer-modifier:*
*...*
unsafe
*operator-modifier:*
*...*
unsafe
*constructor-modifier:*

*...*
unsafe
*destructor-declaration:*
*attributes*$_{opt}$ extern$_{opt}$ unsafe$_{opt}$ ~ *identifier* ( ) *destructor-body*
*attributes*$_{opt}$ unsafe$_{opt}$ extern$_{opt}$ ~ *identifier* ( ) *destructor-body*
*static-constructor-modifiers:*
extern$_{opt}$ unsafe$_{opt}$ *static*
unsafe$_{opt}$ extern$_{opt}$ *static*
extern$_{opt}$ *static* unsafe$_{opt}$
unsafe$_{opt}$ *static* extern$_{opt}$
*static* extern$_{opt}$ unsafe$_{op}$
*static* unsafe$_{opt}$ extern$_{opt}$
*embedded-statement:*
*...*
*unsafe-statement*
*unsafe-statement:*
unsafe *block*
*type:*
*value-type*
*reference-type*
*pointer-type*
*pointer-type:*
*unmanaged-type* *
void *
*unmanaged-type:*
*type*
*primary-no-array-creation-expression:*
*...*
*pointer-member-access*
*pointer-element-access*
*sizeof-expression*
*unary-expression:*
*...*
*pointer-indirection-expression*
*addressof-expression*
*pointer-indirection-expression:*
* *unary-expression*
*pointer-member-access:*
*primary-expression* -> *identifier*
*pointer-element-access:*
*primary-no-array-creation-expression* [ *expression* ]
*addressof-expression:*
& *unary-expression*
*sizeof-expression:*
sizeof ( *unmanaged-type* )
*embedded-statement:*
*...*
*fixed-statement*
*fixed-statement:*
fixed ( *pointer-type* *fixed-pointer-declarators* ) *embedded-statement*
*fixed-pointer-declarators:*
*fixed-pointer-declarator*
*fixed-pointer-declarators* , *fixed-pointer-declarator*
*fixed-pointer-declarator:*
*identifier* = *fixed-pointer-initializer*
*fixed-pointer-initializer:*

& *variable-reference*
*expression*

*variable-initializer:*
*expression*
*array-initializer*
*stackalloc-initializer*

*stackalloc-initializer:*
stackalloc *unmanaged-type* [ *expression* ]