(6.5.9) *equality-expression:*
relational-expression
equality-expression **==** *relational-expression*
equality-expression **!=** *relational-expression*

(6.5.10) *AND-expression:*
equality-expression
AND-expression **&** *equality-expression*

(6.5.11) *exclusive-OR-expression:*
AND-expression
exclusive-OR-expression **^** *AND-expression*

(6.5.12) *inclusive-OR-expression:*
exclusive-OR-expression
inclusive-OR-expression **|** *exclusive-OR-expression*

(6.5.13) *logical-AND-expression:*
inclusive-OR-expression
logical-AND-expression **&&** *inclusive-OR-expression*

(6.5.14) *logical-OR-expression:*
logical-AND-expression
logical-OR-expression **||** *logical-AND-expression*

(6.5.15) *conditional-expression:*
logical-OR-expression
logical-OR-expression **?** *expression* **:** *conditional-expression*

(6.5.16) *assignment-expression:*
conditional-expression
unary-expression assignment-operator assignment-expression

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

(6.5.17) *expression:*
assignment-expression
expression **,** *assignment-expression*

(6.6) *constant-expression:*
conditional-expression

A.2.2 Declarations

(6.7) *declaration:*
declaration-specifiers init-declarator-list$_{\text{opt}}$ **;**
attribute-specifier-sequence declaration-specifiers init-declarator-list **;**
static_assert-declaration
attribute-declaration

(6.7) *declaration-specifiers:*
declaration-specifier attribute-specifier-sequence$_{\text{opt}}$
declaration-specifier declaration-specifiers

(6.7) *declaration-specifier:*
storage-class-specifier
type-specifier-qualifier
function-specifier

(6.7) *init-declarator-list:*
init-declarator
init-declarator-list **,** *init-declarator*

(6.7) *init-declarator:*
> *declarator*
> *declarator* **`=`** *initializer*

(6.7) *attribute-declaration:*
> *attribute-specifier-sequence* **`;`**

(6.7.1) *storage-class-specifier:*
> **`auto`**
> **`constexpr`**
> **`extern`**
> **`register`**
> **`static`**
> **`thread_local`**
> **`typedef`**

(6.7.2) *type-specifier:*
> **`void`**
> **`char`**
> **`short`**
> **`int`**
> **`long`**
> **`float`**
> **`double`**
> **`signed`**
> **`unsigned`**
> **`_BitInt (`** *constant-expression* **`)`**
> **`bool`**
> **`_Complex`**
> **`_Decimal32`**
> **`_Decimal64`**
> **`_Decimal128`**
> *atomic-type-specifier*
> *struct-or-union-specifier*
> *enum-specifier*
> *typedef-name*
> *typeof-specifier*

(6.7.2.1) *struct-or-union-specifier:*
> *struct-or-union attribute-specifier-sequence*$_{\text{opt}}$ *identifier*$_{\text{opt}}$ **`{`** *member-declaration-list* **`}`**
> *struct-or-union attribute-specifier-sequence*$_{\text{opt}}$ *identifier*

(6.7.2.1) *struct-or-union:*
> **`struct`**
> **`union`**

[-2ex]

(6.7.2.1) *member-declaration-list:*
> *member-declaration*
> *member-declaration-list member-declaration*

(6.7.2.1) *member-declaration:*
> *attribute-specifier-sequence*$_{\text{opt}}$ *specifier-qualifier-list member-declarator-list*$_{\text{opt}}$ **`;`**
> *static_assert-declaration*

(6.7.2.1) *specifier-qualifier-list:*
> *type-specifier-qualifier attribute-specifier-sequence*$_{\text{opt}}$
> *type-specifier-qualifier specifier-qualifier-list*

(6.7.2.1) *type-specifier-qualifier:*
type-specifier
type-qualifier
alignment-specifier

(6.7.2.1) *member-declarator-list:*
member-declarator
member-declarator-list **,** *member-declarator*

(6.7.2.1) *member-declarator:*
declarator
declarator$_{\text{opt}}$ **:** *constant-expression*

(6.7.2.2) *enum-specifier:*
enum *attribute-specifier-sequence*$_{\text{opt}}$ *identifier*$_{\text{opt}}$ *enum-type-specifier*$_{\text{opt}}$
{ *enumerator-list* **}**
enum *attribute-specifier-sequence*$_{\text{opt}}$ *identifier*$_{\text{opt}}$ *enum-type-specifier*$_{\text{opt}}$
{ *enumerator-list* **, }**
enum *identifier* *enum-type-specifier*$_{\text{opt}}$

(6.7.2.2) *enumerator-list:*
enumerator
enumerator-list **,** *enumerator*

(6.7.2.2) *enumerator:*
enumeration-constant *attribute-specifier-sequence*$_{\text{opt}}$
enumeration-constant *attribute-specifier-sequence*$_{\text{opt}}$ **=** *constant-expression*
(6.7.2.2) *enum-type-specifier:*
: *specifier-qualifier-list*

(6.7.2.4) *atomic-type-specifier:*
_Atomic (*type-name* **)**

(6.7.2.5) *typeof-specifier:*
typeof (*typeof-specifier-argument* **)**
typeof_unqual (*typeof-specifier-argument* **)**
(6.7.2.5) *typeof-specifier-argument:*
expression
type-name

(6.7.3) *type-qualifier:*
const
restrict
volatile
_Atomic

(6.7.4) *function-specifier:*
inline
_Noreturn
[-7ex]

(6.7.5) *alignment-specifier:*
alignas (*type-name* **)**
alignas (*constant-expression* **)**

(6.7.6) *declarator:*
pointer$_{\text{opt}}$ *direct-declarator*

(6.7.6) *direct-declarator:*
identifier *attribute-specifier-sequence*$_{\text{opt}}$
(*declarator* **)**
array-declarator *attribute-specifier-sequence*$_{\text{opt}}$
function-declarator *attribute-specifier-sequence*$_{\text{opt}}$

(6.7.6) *array-declarator:*
direct-declarator **[** *type-qualifier-list*$_{\text{opt}}$ *assignment-expression*$_{\text{opt}}$ **]**
direct-declarator **[** **`static`** *type-qualifier-list*$_{\text{opt}}$ *assignment-expression* **]**
direct-declarator **[** *type-qualifier-list* **`static`** *assignment-expression* **]**
direct-declarator **[** *type-qualifier-list*$_{\text{opt}}$ ***** **]**

(6.7.6) *function-declarator:*
direct-declarator **(** *parameter-type-list*$_{opt}$ **)**

(6.7.6) *pointer:*
***** *attribute-specifier-sequence*$_{\text{opt}}$ *type-qualifier-list*$_{\text{opt}}$
***** *attribute-specifier-sequence*$_{\text{opt}}$ *type-qualifier-list*$_{\text{opt}}$ *pointer*

(6.7.6) *type-qualifier-list:*
type-qualifier
type-qualifier-list type-qualifier

(6.7.6) *parameter-type-list:*
parameter-list
parameter-list **, ...**
...

(6.7.6) *parameter-list:*
parameter-declaration
parameter-list **,** *parameter-declaration*

(6.7.6) *parameter-declaration:*
attribute-specifier-sequence$_{\text{opt}}$ *declaration-specifiers declarator*
attribute-specifier-sequence$_{\text{opt}}$ *declaration-specifiers abstract-declarator*$_{\text{opt}}$

(6.7.7) *type-name:*
specifier-qualifier-list abstract-declarator$_{\text{opt}}$

(6.7.7) *abstract-declarator:*
pointer
pointer$_{\text{opt}}$ *direct-abstract-declarator*

(6.7.7) *direct-abstract-declarator:*
(*abstract-declarator* **)**
array-abstract-declarator attribute-specifier-sequence$_{\text{opt}}$
function-abstract-declarator attribute-specifier-sequence$_{\text{opt}}$

(6.7.7) *array-abstract-declarator:*
direct-abstract-declarator$_{\text{opt}}$ **[** *type-qualifier-list*$_{\text{opt}}$ *assignment-expression*$_{\text{opt}}$ **]**
direct-abstract-declarator$_{\text{opt}}$ **[** **`static`** *type-qualifier-list*$_{\text{opt}}$ *assignment-expression* **]**
direct-abstract-declarator$_{\text{opt}}$ **[** *type-qualifier-list* **`static`** *assignment-expression* **]**
direct-abstract-declarator$_{\text{opt}}$ **[*]**

(6.7.7) *function-abstract-declarator:*
direct-abstract-declarator$_{\text{opt}}$ **(** *parameter-type-list*$_{\text{opt}}$ **)**

(6.7.8) *typedef-name:*
identifier

(6.7.10) *braced-initializer:*
{ }
{ *initializer-list* **}**
{ *initializer-list* **, }**

(6.7.10) *initializer:*
assignment-expression
braced-initializer

(6.7.10) *initializer-list:*
designation$_{\text{opt}}$ *initializer*
initializer-list **,** *designation*$_{\text{opt}}$ *initializer*

(6.7.10) *designation:*
designator-list **=**

(6.7.10) *designator-list:*
designator
designator-list designator

(6.7.10) *designator:*
[*constant-expression* **]**
. *identifier*

(6.7.11) *static_assert-declaration:*
static_assert (*constant-expression* **,** *string-literal* **) ;**
static_assert (*constant-expression* **) ;**

(6.7.12.1) *attribute-specifier-sequence:*
attribute-specifier-sequence$_{\text{opt}}$ *attribute-specifier*

(6.7.12.1) *attribute-specifier:*
[[*attribute-list* **]]**

(6.7.12.1) *attribute-list:*
attribute$_{\text{opt}}$
attribute-list **,** *attribute*$_{\text{opt}}$

(6.7.12.1) *attribute:*
attribute-token attribute-argument-clause$_{\text{opt}}$

(6.7.12.1) *attribute-token:*
standard-attribute
attribute-prefixed-token

(6.7.12.1) *standard-attribute:*
identifier

(6.7.12.1) *attribute-prefixed-token:*
attribute-prefix **::** *identifier*

(6.7.12.1) *attribute-prefix:*
identifier

(6.7.12.1) *attribute-argument-clause:*
(*balanced-token-sequence*$_{\text{opt}}$ **)**

(6.7.12.1) *balanced-token-sequence:*
balanced-token
balanced-token-sequence balanced-token

(6.7.12.1) *balanced-token:*
(*balanced-token-sequence*$_{\text{opt}}$ **)**
[*balanced-token-sequence*$_{\text{opt}}$ **]**
{ *balanced-token-sequence*$_{\text{opt}}$ **}**
any token other than a parenthesis, a bracket, or a brace

A.2.3 Statements

(6.8) *statement:*
labeled-statement
unlabeled-statement

(6.8) *unlabeled-statement:*
expression-statement
attribute-specifier-sequence$_{\text{opt}}$ *primary-block*
attribute-specifier-sequence$_{\text{opt}}$ *jump-statement*