Annex A
(informative)
Language syntax summary

1 **NOTE 1** The notation is described in 6.1.

A.1 Lexical grammar

A.1.1 Lexical elements

(6.4) *token:*
> *keyword*
> *identifier*
> *constant*
> *string-literal*
> *punctuator*

(6.4) *preprocessing-token:*
> *header-name*
> *identifier*
> *pp-number*
> *character-constant*
> *string-literal*
> *punctuator*
> each universal-character-name that cannot be one of the above
> each non-white-space character that cannot be one of the above

A.1.2 Keywords

(6.4.1) *keyword:* one of

`alignas`	`enum`	`short`	`void`
`alignof`	`extern`	`signed`	`volatile`
`auto`	`false`	`sizeof`	`while`
`bool`	`float`	`static`	`_Atomic`
`break`	`for`	`static_assert`	`_BitInt`
`case`	`goto`	`struct`	`_Complex`
`char`	`if`	`switch`	`_Decimal128`
`const`	`inline`	`thread_local`	`_Decimal32`
`constexpr`	`int`	`true`	`_Decimal64`
`continue`	`long`	`typedef`	`_Generic`
`default`	`nullptr`	`typeof`	`_Imaginary`
`do`	`register`	`typeof_unqual`	`_Noreturn`
`double`	`restrict`	`union`	
`else`	`return`	`unsigned`	

A.1.3 Identifiers

(6.4.2.1) *identifier:*
> *identifier-start*
> *identifier identifier-continue*

(6.4.2.1) *identifier-start:*
> *nondigit*
> XID_Start character
> universal-character-name of class XID_Start

(6.4.2.1) *identifier-continue:*
> *digit*
> *nondigit*
> XID_Continue character
> universal-character-name of class XID_Continue

(6.4.2.1) *nondigit:* one of

```
_ a b c d e f g h i j k l m
  n o p q r s t u v w x y z
  A B C D E F G H I J K L M
  N O P Q R S T U V W X Y Z
```

(6.4.2.1) *digit:* one of

```
0 1 2 3 4 5 6 7 8 9
```

A.1.4 Universal character names

(6.4.3) *universal-character-name:*
- `\u` *hex-quad*
- `\U` *hex-quad hex-quad*

(6.4.3) *hex-quad:*
- *hexadecimal-digit hexadecimal-digit hexadecimal-digit hexadecimal-digit*

A.1.5 Constants

(6.4.4) *constant:*
- *integer-constant*
- *floating-constant*
- *enumeration-constant*
- *character-constant*
- *predefined-constant*

(6.4.4.1) *integer-constant:*
- *decimal-constant integer-suffix*$_{\text{opt}}$
- *octal-constant integer-suffix*$_{\text{opt}}$
- *hexadecimal-constant integer-suffix*$_{\text{opt}}$
- *binary-constant integer-suffix*$_{\text{opt}}$

(6.4.4.1) *decimal-constant:*
- *nonzero-digit*
- *decimal-constant* `'`$_{\text{opt}}$ *digit*

(6.4.4.1) *octal-constant:*
- `0`
- *octal-constant* `'`$_{\text{opt}}$ *octal-digit*

(6.4.4.1) *hexadecimal-constant:*
- *hexadecimal-prefix hexadecimal-digit-sequence*

(6.4.4.1) *binary-constant:*
- *binary-prefix binary-digit*
- *binary-constant* `'`$_{\text{opt}}$ *binary-digit*

(6.4.4.1) *hexadecimal-prefix:* one of

```
0x 0X
```

(6.4.4.1) *binary-prefix:* one of

```
0b 0B
```

(6.4.4.1) *nonzero-digit:* one of

```
1 2 3 4 5 6 7 8 9
```

(6.4.4.1) *octal-digit:* one of

```
0 1 2 3 4 5 6 7
```

hexadecimal-digit-sequence:
- *hexadecimal-digit*
- *hexadecimal-digit-sequence* `'`$_{\text{opt}}$ *hexadecimal-digit*

(6.4.4.1) *hexadecimal-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`

(6.4.4.1) *binary-digit:* one of

    `0 1`

(6.4.4.1) *integer-suffix:*

    unsigned-suffix long-suffix$_{\text{opt}}$
    unsigned-suffix long-long-suffix
    unsigned-suffix bit-precise-int-suffix
    long-suffix unsigned-suffix$_{\text{opt}}$
    long-long-suffix unsigned-suffix$_{\text{opt}}$
    bit-precise-int-suffix unsigned-suffix$_{\text{opt}}$

(6.4.4.1) *bit-precise-int-suffix:* one of

    `wb WB`

(6.4.4.1) *unsigned-suffix:* one of

    `u U`

(6.4.4.1) *long-suffix:* one of

    `l L`

(6.4.4.1) *long-long-suffix:* one of

    `ll LL`

(6.4.4.2) *floating-constant:*

    decimal-floating-constant
    hexadecimal-floating-constant

(6.4.4.2) *decimal-floating-constant:*

    fractional-constant exponent-part$_{\text{opt}}$ *floating-suffix*$_{\text{opt}}$
    digit-sequence exponent-part floating-suffix$_{\text{opt}}$

(6.4.4.2) *hexadecimal-floating-constant:*

    hexadecimal-prefix hexadecimal-fractional-constant
        binary-exponent-part floating-suffix$_{\text{opt}}$
    hexadecimal-prefix hexadecimal-digit-sequence
        binary-exponent-part floating-suffix$_{\text{opt}}$

(6.4.4.2) *fractional-constant:*

    digit-sequence$_{\text{opt}}$ **`.`** *digit-sequence*
    digit-sequence **`.`**

(6.4.4.2) *exponent-part:*

    `e` *sign*$_{\text{opt}}$ *digit-sequence*
    `E` *sign*$_{\text{opt}}$ *digit-sequence*

(6.4.4.2) *sign:* one of

    `+ -`

(6.4.4.2) *digit-sequence:*

    digit
    digit-sequence **`'`**$_{\text{opt}}$ *digit*

(6.4.4.2) *hexadecimal-fractional-constant:*

    hexadecimal-digit-sequence$_{\text{opt}}$ **`.`** *hexadecimal-digit-sequence*
    hexadecimal-digit-sequence **`.`**

(6.4.4.2) *binary-exponent-part:*

    `p` *sign*$_{\text{opt}}$ *digit-sequence*
    `P` *sign*$_{\text{opt}}$ *digit-sequence*

(6.4.4.2) *floating-suffix:* one of

`f l F L df dd dl DF DD DL`

(6.4.4.3) *enumeration-constant:*

identifier

(6.4.4.4) *character-constant:*

encoding-prefix$_{\text{opt}}$ **`'`** *c-char-sequence* **`'`**

(6.4.4.4) *encoding-prefix:* one of

`u8` **`u`** **`U`** **`L`**

(6.4.4.4) *c-char-sequence:*

c-char
c-char-sequence c-char

(6.4.4.4) *c-char:*

any member of the source character set except
the single-quote `'`, backslash `\`, or new-line character
escape-sequence

(6.4.4.4) *escape-sequence:*

simple-escape-sequence
octal-escape-sequence
hexadecimal-escape-sequence
universal-character-name

(6.4.4.4) *simple-escape-sequence:* one of

`\' \" \? \\`
`\a \b \f \n \r \t \v`

(6.4.4.4) *octal-escape-sequence:*

`\` *octal-digit*
`\` *octal-digit octal-digit*
`\` *octal-digit octal-digit octal-digit*

(6.4.4.4) *hexadecimal-escape-sequence:*

`\x` *hexadecimal-digit*
hexadecimal-escape-sequence hexadecimal-digit

(6.4.4.5) *predefined-constant:*

`false`
`true`
`nullptr`

A.1.6 String literals

(6.4.5) *string-literal:*

encoding-prefix$_{\text{opt}}$ **`"`** *s-char-sequence*$_{\text{opt}}$ **`"`**

(6.4.5) *s-char-sequence:*

s-char
s-char-sequence s-char

(6.4.5) *s-char:*

any member of the source character set except
the double-quote `"`, backslash `\`, or new-line character
escape-sequence

A.1.7 Punctuators

(6.4.6) *punctuator:* one of

```
[   ]   (  )   {   }   .    ->
++   --   &   *   +   -   ~   !
/   %   <<   >>   <   >   <=   >=   ==   !=   ^   |   &&   ||
?   :   ::   ;   ...
=   *=   /=   %=   +=   -=   <<=   >>=   &=   ^=   |=
,   #   ##
<:   :>   <%   %>   %:   %:%:
```

A.1.8 Header names

(6.4.7) *header-name:*
  < *h-char-sequence* **>**
  " *q-char-sequence* **"**

(6.4.7) *h-char-sequence:*
  h-char
  h-char-sequence h-char

(6.4.7) *h-char:*
  any member of the source character set except
    the new-line character and >

(6.4.7) *q-char-sequence:*
  q-char
  q-char-sequence q-char

(6.4.7) *q-char:*
  any member of the source character set except
    the new-line character and **"**

A.1.9 Preprocessing numbers

(6.4.8) *pp-number:*
  digit
  . *digit*
  pp-number identifier-continue
  pp-number **'** *digit*
  pp-number **'** *nondigit*
  pp-number **e** *sign*
  pp-number **E** *sign*
  pp-number **p** *sign*
  pp-number **P** *sign*
  pp-number **.**

A.2 Phrase structure grammar

A.2.1 Expressions

(6.5.1) *primary-expression:*
  identifier
  constant
  string-literal
  (*expression* **)**
  generic-selection

(6.5.1.1) *generic-selection:*
  _Generic (*assignment-expression* **,** *generic-assoc-list* **)**

(6.5.1.1) *generic-assoc-list:*
  generic-association
  generic-assoc-list **,** *generic-association*

(6.5.1.1) *generic-association:*
  type-name **:** *assignment-expression*
  `default` **:** *assignment-expression*

(6.5.2) *postfix-expression:*
  primary-expression
  postfix-expression **[** *expression* **]**
  postfix-expression **(** *argument-expression-list*$_{\text{opt}}$ **)**
  postfix-expression **.** *identifier*
  postfix-expression **->** *identifier*
  postfix-expression **++**
  postfix-expression **--**
  compound-literal

(6.5.2) *argument-expression-list:*
  assignment-expression
  argument-expression-list **,** *assignment-expression*

(6.5.2.5) *compound-literal:*
  (*storage-class-specifiers*$_{\text{opt}}$ *type-name* **)** *braced-initializer*

(6.5.2.5) *storage-class-specifiers:*
  storage-class-specifier
  storage-class-specifiers storage-class-specifier

(6.5.3) *unary-expression:*
  postfix-expression
  ++ *unary-expression*
  -- *unary-expression*
  unary-operator cast-expression
  `sizeof` *unary-expression*
  `sizeof` **(** *type-name* **)**
  `alignof` **(** *type-name* **)**

(6.5.3) *unary-operator:* one of
  & * + - ~ !

(6.5.4) *cast-expression:*
  unary-expression
  (*type-name* **)** *cast-expression*

(6.5.5) *multiplicative-expression:*
  cast-expression
  multiplicative-expression ***** *cast-expression*
  multiplicative-expression **/** *cast-expression*
  multiplicative-expression **%** *cast-expression*

(6.5.6) *additive-expression:*
  multiplicative-expression
  additive-expression **+** *multiplicative-expression*
  additive-expression **-** *multiplicative-expression*

(6.5.7) *shift-expression:*
  additive-expression
  shift-expression **<<** *additive-expression*
  shift-expression **>>** *additive-expression*

(6.5.8) *relational-expression:*
  shift-expression
  relational-expression **<** *shift-expression*
  relational-expression **>** *shift-expression*
  relational-expression **<=** *shift-expression*
  relational-expression **>=** *shift-expression*

(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*

(6.8) *primary-block:*
> *compound-statement*
> *selection-statement*
> *iteration-statement*

(6.8) *secondary-block:*
> *statement*

(6.8.1) *label:*
> *attribute-specifier-sequence*$_{\text{opt}}$ *identifier* **`:`**
> *attribute-specifier-sequence*$_{\text{opt}}$ **`case`** *constant-expression* **`:`**
> *attribute-specifier-sequence*$_{\text{opt}}$ **`default`** **`:`**

(6.8.1) *labeled-statement:*
> *label statement*

(6.8.2) *compound-statement:*
> **`{`** *block-item-list*$_{\text{opt}}$ **`}`**

(6.8.2) *block-item-list:*
> *block-item*
> *block-item-list block-item*

(6.8.2) *block-item:*
> *declaration*
> *unlabeled-statement*
> *label*

(6.8.3) *expression-statement:*
> *expression*$_{\text{opt}}$ **`;`**
> *attribute-specifier-sequence expression* **`;`**

[-6ex]

(6.8.4) *selection-statement:*
> **`if`** **`(`** *expression* **`)`** *secondary-block*
> **`if`** **`(`** *expression* **`)`** *secondary-block* **`else`** *secondary-block*
> **`switch`** **`(`** *expression* **`)`** *secondary-block*

[-6ex]

(6.8.5) *iteration-statement:*
> **`while`** **`(`** *expression* **`)`** *secondary-block*
> **`do`** *secondary-block* **`while`** **`(`** *expression* **`)`** **`;`**
> **`for`** **`(`** *expression*$_{\text{opt}}$ **`;`** *expression*$_{\text{opt}}$ **`;`** *expression*$_{\text{opt}}$ **`)`** *secondary-block*
> **`for`** **`(`** *declaration expression*$_{\text{opt}}$ **`;`** *expression*$_{\text{opt}}$ **`)`** *secondary-block*

[-6ex]

(6.8.6) *jump-statement:*
> **`goto`** *identifier* **`;`**
> **`continue`** **`;`**
> **`break`** **`;`**
> **`return`** *expression*$_{\text{opt}}$ **`;`**

[-6ex]

A.2.4 External definitions

(6.9) *translation-unit:*
> *external-declaration*
> *translation-unit external-declaration*

(6.9) *external-declaration:*
> *function-definition*
> *declaration*

(6.9.1) *function-definition:*
    attribute-specifier-sequence$_{\text{opt}}$ *declaration-specifiers declarator function-body*

(6.9.1) *function-body:*
    compound-statement

A.3 Preprocessing directives

(6.10) *preprocessing-file:*
    group$_{\text{opt}}$

(6.10) *group:*
    group-part
    group group-part

(6.10) *group-part:*
    if-section
    control-line
    text-line
    # *non-directive*

(6.10) *if-section:*
    if-group elif-groups$_{\text{opt}}$ *else-group*$_{\text{opt}}$ *endif-line*

(6.10) *if-group:*
    # if *constant-expression new-line group*$_{\text{opt}}$
    # ifdef *identifier new-line group*$_{\text{opt}}$
    # ifndef *identifier new-line group*$_{\text{opt}}$

(6.10) *elif-groups:*
    elif-group
    elif-groups elif-group

(6.10) *elif-group:*
    # elif *constant-expression new-line group*$_{\text{opt}}$
    # elifdef *identifier new-line group*$_{\text{opt}}$
    # elifndef *identifier new-line group*$_{\text{opt}}$

(6.10) *else-group:*
    # else *new-line group*$_{\text{opt}}$

(6.10) *endif-line:*
    # endif *new-line*

(6.10) *control-line:*
    # include *pp-tokens new-line*
    # embed *pp-tokens new-line*
    # define *identifier replacement-list new-line*
    # define *identifier lparen identifier-list*$_{\text{opt}}$ **)** *replacement-list new-line*
    # define *identifier lparen* **...)** *replacement-list new-line*
    # define *identifier lparen identifier-list* **, ...)** *replacement-list new-line*
    # undef *identifier new-line*
    # line *pp-tokens new-line*
    # error *pp-tokens*$_{\text{opt}}$ *new-line*
    # warning *pp-tokens*$_{\text{opt}}$ *new-line*
    # pragma *pp-tokens*$_{\text{opt}}$ *new-line*
    # *new-line*

(6.10) *text-line:*
    pp-tokens$_{\text{opt}}$ *new-line*

(6.10) *non-directive:*
    pp-tokens new-line

(6.10) *lparen:*

a **(** character not immediately preceded by white space

(6.10) *replacement-list:*

pp-tokens$_{\text{opt}}$

(6.10) *pp-tokens:*

preprocessing-token
pp-tokens preprocessing-token

(6.10) *new-line:*

the new-line character

(6.10) *identifier-list:*

identifier
identifier-list **,** *identifier*

(6.10) *pp-parameter:*

pp-parameter-name pp-parameter-clause$_{\text{opt}}$

(6.10) *pp-parameter-name:*

pp-standard-parameter
pp-prefixed-parameter

(6.10) *pp-standard-parameter:*

identifier

(6.10) *pp-prefixed-parameter:*

identifier **::** *identifier*

(6.10) *pp-parameter-clause:*

(*pp-balanced-token-sequence*$_{\text{opt}}$ **)**

(6.10) *pp-balanced-token-sequence:*

pp-balanced-token
pp-balanced-token-sequence pp-balanced-token

(6.10) *pp-balanced-token:*

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

(6.10) *embed-parameter-sequence:*

pp-parameter
embed-parameter-sequence pp-parameter

defined-macro-expression:
  `defined` *identifier*
  `defined (` *identifier* **`)`**

h-preprocessing-token:
  any *preprocessing-token* other than **`>`**

h-pp-tokens:
  h-preprocessing-token
  h-pp-tokens h-preprocessing-token

header-name-tokens:
  string-literal
  `<` *h-pp-tokens* **`>`**

has-include-expression:
  `__has_include (` *header-name* **`)`**
  `__has_include (` *header-name-tokens* **`)`**

has-embed-expression:
  `__has_embed (` *header-name embed-parameter-sequence*$_{\text{opt}}$ **`)`**
  `__has_embed (` *header-name-tokens pp-balanced-token-sequence*$_{\text{opt}}$ **`)`**

has-c-attribute-express:
  `__has_c_attribute (` *pp-tokens* **`)`**

va-opt-replacement:
  `__VA_OPT__ (` *pp-tokens*$_{\text{opt}}$ **`)`**

(6.10.7) *standard-pragma:*
  `# pragma` `STDC FP_CONTRACT` *on-off-switch*
  `# pragma` `STDC FENV_ACCESS` *on-off-switch*
  `# pragma` `STDC FENV_DEC_ROUND` *dec-direction*
  `# pragma` `STDC FENV_ROUND` *direction*
  `# pragma` `STDC CX_LIMITED_RANGE` *on-off-switch*

(6.10.7) *on-off-switch:* one of
  `ON` `OFF` `DEFAULT`

(6.10.7) *direction:* one of
  `FE_DOWNWARD` `FE_TONEAREST` `FE_TONEARESTFROMZERO`
  `FE_TOWARDZERO` `FE_UPWARD` `FE_DYNAMIC`

(6.10.7) *dec-direction:* one of
  `FE_DEC_DOWNWARD` `FE_DEC_TONEAREST` `FE_DEC_TONEARESTFROMZERO`
  `FE_DEC_TOWARDZERO` `FE_DEC_UPWARD` `FE_DEC_DYNAMIC`

A.4 Floating-point subject sequence

A.4.1 NaN `char` sequence

(7.24.1.5) *n-char-sequence*:
  digit
  nondigit
  n-char-sequence digit
  n-char-sequence nondigit

A.4.2 NaN `wchar_t` sequence

(7.31.4.1.2) *n-wchar-sequence*:
  digit
  nondigit
  n-wchar-sequence digit
  n-wchar-sequence nondigit

A.5 Decimal floating-point subject sequence

A.5.1 NaN decimal `char` sequence

(7.24.1.6) *d-char-sequence*:
  digit
  nondigit
  d-char-sequence digit
  d-char-sequence nondigit

A.5.2 NaN decimal `wchar_t` sequence

(7.31.4.1.3) *d-wchar-sequence*:
  digit
  nondigit
  d-wchar-sequence digit
  d-wchar-sequence nondigit