

Syntax for Ada 2022

The following syntax rules are taken from Annex P of the *ARM*. The rules have been reordered to correspond to the order of introduction of the topics in this book but individual rules have not been changed.

It should be noted that the rules for the construction of lexical elements, which are under the subheading of Chapter 5, have a slightly different status from the other rules since spaces and newlines may be freely inserted between lexical elements but not within lexical elements.

The rules have been sequentially numbered for ease of reference; an index to them will be found in Section 2. Note that the symbols [] and | enclosed in single quotes stand for themselves and are not metasympols. Examples are in rule 75 (*discrete_choice_list*) and in rule 111 (*null_array_aggregate*).

1 Syntax rules

Chapter 5

- 1 identifier ::= identifier_start {identifier_start | identifier_extend}
- 2 identifier_start ::= letter_uppercase | letter_lowercase | letter_titlecase
| letter_modifier | letter_other | number_letter
- 3 identifier_extend ::= mark_nonspacing | mark_spacing_combining
| number_decimal | punctuation_connector
- 4 numeric_literal ::= decimal_literal | based_literal
- 5 decimal_literal ::= numeral [. numeral] [exponent]
- 6 numeral ::= digit {[underline] digit}
- 7 exponent ::= E [+] numeral | E - numeral
- 8 digit ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
- 9 based_literal ::= base # based_numeral [. based_numeral] # [exponent]
- 10 base ::= numeral
- 11 based_numeral ::= extended_digit {[underline] extended_digit}
- 12 extended_digit ::= digit | A | B | C | D | E | F
- 13 character_literal ::= 'graphic_character'


```

28 type_declaration ::=
    full_type_declaration      | incomplete_type_declaration
    | private_type_declaration | private_extension_declaration
29 full_type_declaration ::=
    type defining_identifier [known_discriminant_part]
                                is type_definition [aspect_specification];
    | task_type_declaration
    | protected_type_declaration
30 type_definition ::=
    enumeration_type_definition | integer_type_definition
    | real_type_definition      | array_type_definition
    | record_type_definition    | access_type_definition
    | derived_type_definition   | interface_type_definition
31 subtype_declaration ::= subtype defining_identifier is subtype_indication
                                [aspect_specification];
32 subtype_indication ::= [null_exclusion] subtype_mark [constraint]
33 subtype_mark ::= subtype_name
34 constraint ::= scalar_constraint | composite_constraint
35 scalar_constraint ::= range_constraint | digits_constraint | delta_constraint
36 composite_constraint ::= index_constraint | discriminant_constraint
37 range_constraint ::= range range
38 range ::= range_attribute_reference | simple_expression .. simple_expression
39 enumeration_type_definition ::=
    (enumeration_literal_specification {, enumeration_literal_specification})
40 enumeration_literal_specification ::= defining_identifier | defining_character_literal
41 defining_character_literal ::= character_literal
42 name ::= direct_name | explicit_dereference | indexed_component | slice
    | selected_component | attribute_reference
    | type_conversion | function_call | character_literal
    | qualified_expression | generalized_reference | generalized_indexing
    | target_name
43 direct_name ::= identifier | operator_symbol
44 prefix ::= name | implicit_dereference
45 explicit_dereference ::= name . all
46 implicit_dereference ::= name
47 attribute_reference ::= prefix ' attribute_designator | reduction_attribute_reference
48 range_attribute_reference ::= prefix ' range_attribute_designator
49 attribute_designator ::= identifier [(static_expression)] | Access | Delta | Digits | Mod
50 range_attribute_designator ::= Range [(static_expression)]

```

4 Syntax

- 51 expression ::=
 relation {**and** relation}
 | relation {**and then** relation}
 | relation {**or** relation}
 | relation {**or else** relation}
 | relation {**xor** relation}
- 52 relation ::=
 simple_expression [relational_operator simple_expression]
 | *tested*_simple_expression [**not**] **in** membership_choice_list
 | raise_expression
- 53 simple_expression ::= [unary_adding_operator] term {binary_adding_operator term}
- 54 term ::= factor {multiplying_operator factor}
- 55 factor ::= primary [****** primary] | **abs** primary | **not** primary
- 56 primary ::= numeric_literal | **null** | string_literal | aggregate | name | allocator
 | (expression) | (conditional_expression) | (quantified_expression)
 | (declare_expression)
- 57 logical_operator ::= **and** | **or** | **xor**
- 58 relational_operator ::= = | /= | < | <= | > | >=
- 59 binary_adding_operator ::= + | - | &
- 60 unary_adding_operator ::= + | -
- 61 multiplying_operator ::= * | / | **mod** | **rem**
- 62 highest_precedence_operator ::= ****** | **abs** | **not**
- 63 type_conversion ::= subtype_mark (expression) | subtype_mark (name)
- 64 qualified_expression ::= subtype_mark ' (expression) | subtype_mark ' aggregate

Chapter 7

- 65 sequence_of_statements ::= statement {statement} {label}
- 66 statement ::= {label} simple_statement | {label} compound_statement
- 67 simple_statement ::=
 null_statement | assignment_statement
 | exit_statement | goto_statement
 | procedure_call_statement | simple_return_statement
 | entry_call_statement | requeue_statement
 | delay_statement | abort_statement
 | raise_statement | code_statement
- 68 compound_statement ::=
 if_statement | case_statement
 | loop_statement | block_statement
 | extended_return_statement | parallel_block_statement
 | accept_statement | select_statement
- 69 label ::= <<label_statement_identifier>>
- 70 null_statement ::= **null**;

6 Syntax

```

86 procedural_iterator ::=
    iterator_parameter_specification of iterator_procedure_call [iterator_filter]
87 iterator_parameter_specification ::= formal_part
    | (defining_identifier {, defining_identifier})
88 iterator_procedure_call ::=
    procedure_name | procedure_prefix iterator_actual_parameter_part
89 iterator_actual_parameter_part ::=
    (iterator_parameter_association {, iterator_parameter_association})
90 iterator_parameter_association ::=
    parameter_association | parameter_association_with_box
91 parameter_association_with_box ::= [formal_parameter_selector_name => ] <>
92 exit_statement ::= exit [loop_name] [when condition];
93 goto_statement ::= goto label_name;

```

Chapter 8

```

94 array_type_definition ::= unconstrained_array_definition | constrained_array_definition
95 unconstrained_array_definition ::=
    array (index_subtype_definition
    {, index_subtype_definition}) of component_definition
96 constrained_array_definition ::=
    array (discrete_subtype_definition
    {, discrete_subtype_definition}) of component_definition
97 index_subtype_definition ::= subtype_mark range <>
98 discrete_subtype_definition ::= discrete_subtype_indication | range
99 component_definition ::= [aliased] subtype_indication | [aliased] access_definition
100 index_constraint ::= (discrete_range {, discrete_range})
101 discrete_range ::= discrete_subtype_indication | range
102 indexed_component ::= prefix (expression {, expression})
103 slice ::= prefix (discrete_range)
104 aggregate ::= record_aggregate | extension_aggregate | array_aggregate
    | delta_aggregate | container_aggregate
105 record_aggregate ::= (record_component_association_list)
106 record_component_association_list ::=
    record_component_association {, record_component_association}
    | null record
107 record_component_association ::=
    [component_choice_list =>] expression
    | component_choice_list => <>
108 component_choice_list ::=
    component_selector_name { '|' component_selector_name }
    | others

```

```

109 array_aggregate ::= positional_array_aggregate | null_array_aggregate
    | named_array_aggregate
110 positional_array_aggregate ::=
    (expression , expression { , expression })
    | (expression { , expression } , others => expression)
    | (expression { , expression } , others => <>)
    | '[' expression { , expression } [, others => expression] ']'
    | '[' expression { , expression } , others => <> ']'
111 null_array_aggregate ::= '[' ']'
112 named_array_aggregate ::=
    (array_component_association_list) | '[' array_component_association_list ']'
113 array_component_association_list ::=
    array_component_association { , array_component_association }
114 array_component_association ::=
    discrete_choice_list => expression | discrete_choice_list => <>
    | iterated_component_association
115 iterated_component_association ::=
    for defining_identifier in discrete_choice_list => expression
    | for iterator_specification => expression
116 delta_aggregate ::= record_delta_aggregate | array_delta_aggregate
117 record_delta_aggregate ::=
    (base_expression with delta record_component_association_list)
118 array_delta_aggregate ::=
    (base_expression with delta array_component_association_list)
    | '[' base_expression with delta array_component_association_list ']'
119 record_type_definition ::= [[abstract] tagged] [limited] record_definition
120 record_definition ::=
    record
    component_list
    end record [record_identifier]
    | null record
121 component_list ::=
    component_item { component_item }
    | { component_item } variant_part
    | null;
122 component_item ::= component_declaration | aspect_clause
123 component_declaration ::= defining_identifier_list :
    component_definition [= default_expression] [aspect_specification];
124 default_expression ::= expression
125 selected_component ::= prefix . selector_name
126 selector_name ::= identifier | character_literal | operator_symbol

```

Chapter 9

```

127 membership_choice_list ::= membership_choice { '[' membership_choice }

```

8 Syntax

```

128 membership_choice ::= choice_simple_expression | range | subtype_mark
129 conditional_expression ::= if_expression | case_expression
130 if_expression ::=
        if condition then dependent_expression
        {elsif condition then dependent_expression}
        [else dependent_expression]
131 case_expression ::=
        case selecting_expression is
        case_expression_alternative
        {, case_expression_alternative}
132 case_expression_alternative ::=
        when discrete_choice_list => dependent_expression
133 quantified_expression ::=
        for quantifier loop_parameter_specification => predicate
        | for quantifier iterator_specification => predicate
134 quantifier ::= all | some
135 predicate ::= boolean_expression
136 declare_expression ::= declare {declare_item} begin body_expression
137 declare_item ::= object_declaration | object_renaming_declaration
138 reduction_attribute_reference ::=
        value_sequence ' reduction_attribute_designator
        | prefix ' reduction_attribute_designator
139 value_sequence ::= '[' [parallel [(chunk_specification)] [aspect_specification]]
        iterated_element_association ']'
140 reduction_attribute_designator ::= reduction_identifier(reduction_specification)
141 reduction_specification ::= reducer_name,initial_value_expression

```

Chapter 10

```

142 subprogram_declaration ::=
        [overriding_indicator]
        subprogram_specification [aspect_specification];
143 subprogram_specification ::= procedure_specification | function_specification
144 procedure_specification ::= procedure defining_program_unit_name parameter_profile
145 function_specification ::= function defining_designator parameter_and_result_profile
146 parameter_profile ::= [formal_part]
147 parameter_and_result_profile ::=
        [formal_part] return [null_exclusion] subtype_mark
        | [formal_part] return access_definition
148 designator ::= [parent_unit_name . ] identifier | operator_symbol
149 defining_designator ::= defining_program_unit_name | defining_operator_symbol
150 defining_program_unit_name ::= [parent_unit_name . ] defining_identifier
151 parent_unit_name ::= name

```



```

152 operator_symbol ::= string_literal
153 defining_operator_symbol ::= operator_symbol
154 formal_part ::= (parameter_specification {; parameter_specification})
155 parameter_specification ::=
    defining_identifier_list : [aliased] mode [null_exclusion]
                                subtype_mark [:= default_expression] [aspect_specification]
    | defining_identifier_list : access_definition
                                [:= default_expression] [aspect_specification]

156 mode ::= [in] | in out | out
157 subprogram_body ::=
    [overriding_indicator]
    subprogram_specification
    [aspect_specification] is
    declarative_part
    begin
    handled_sequence_of_statements
    end [designator];
158 procedure_call_statement ::=
    procedure_name; | procedure_prefix actual_parameter_part;
159 function_call ::=
    function_name | function_prefix actual_parameter_part
160 actual_parameter_part ::=
    (parameter_association {, parameter_association})
161 parameter_association ::=
    [formal_parameter_selector_name =>] explicit_actual_parameter
162 explicit_actual_parameter ::= expression | variable_name
163 simple_return_statement ::= return [expression];
164 extended_return_object_declaration ::=
    defining_identifier : [aliased] [constant] return_subtype_indication
                                [:= expression][aspect_specification]
165 extended_return_statement ::=
    return extended_return_object_declaration [do
    handled_sequence_of_statements
    end return];
166 return_subtype_indication ::= subtype_indication | access_definition
167 expression_function_declaration ::=
    [overriding_indicator]
    function_specification is
    (expression)
    [aspect_specification];
    | [overriding_indicator]
    function_specification is
    aggregate
    [aspect_specification];

```

10 Syntax

Chapter 11

```

168 access_type_definition ::=
    [null_exclusion] access_to_object_definition
    | [null_exclusion] access_to_subprogram_definition
169 access_to_object_definition ::=
    access [general_access_modifier] subtype_indication
170 general_access_modifier ::= all | constant
171 access_to_subprogram_definition ::=
    access [protected] procedure parameter_profile
    | access [protected] function parameter_and_result_profile
172 null_exclusion ::= not null
173 access_definition ::=
    [null_exclusion] access [constant] subtype_mark
    | [null_exclusion] access [protected] procedure parameter_profile
    | [null_exclusion] access [protected] function parameter_and_result_profile
174 incomplete_type_declaration ::=
    type defining_identifier [discriminant_part] [is tagged];
175 allocator ::=
    new [subpool_specification] subtype_indication
    | new [subpool_specification] qualified_expression
176 subpool_specification ::= (subpool_handle_name)

```

Chapter 12

```

177 package_declaration ::= package_specification;
178 package_specification ::=
    package defining_program_unit_name
    [aspect_specification] is
    {basic_declarative_item}
    [private
    {basic_declarative_item}]
    end [[parent_unit_name . ] identifier]
179 package_body ::=
    package body defining_program_unit_name
    [aspect_specification] is
    declarative_part
    [begin
    handled_sequence_of_statements]
    end [[parent_unit_name . ] identifier];
180 declarative_part ::= {declarative_item}
181 declarative_item ::= basic_declarative_item | body
182 basic_declarative_item ::= basic_declaration | aspect_clause | use_clause
183 body ::= proper_body | body_stub
184 proper_body ::=
    subprogram_body          | package_body
    | task_body               | protected_body

```

```

185 use_clause ::= use_package_clause | use_type_clause
186 use_package_clause ::= use package_name {, package_name};
187 use_type_clause ::= use [all] type subtype_mark {, subtype_mark};
188 private_type_declaration ::= type defining_identifier [discriminant_part] is
    [[abstract] tagged] [limited] private
    [aspect_specification];
189 derived_type_definition ::=
    [abstract] [limited] new parent_subtype_indication
    [[and interface_list] record_extension_part]
190 abstract_subprogram_declaration ::=
    [overriding_indicator]
    subprogram_specification is abstract
    [aspect_specification];
191 null_procedure_declaration ::=
    [overriding_indicator]
    procedure_specification is null
    [aspect_specification];
192 overriding_indicator ::= [not] overriding

```

Chapter 13

```

193 compilation ::= {compilation_unit}
194 compilation_unit ::= context_clause library_item | context_clause subunit
195 library_item ::=
    [private] library_unit_declaration | library_unit_body
    | [private] library_unit_renaming_declaration
196 library_unit_declaration ::=
    subprogram_declaration | package_declaration
    | generic_declaration | generic_instantiation
197 library_unit_renaming_declaration ::=
    package_renaming_declaration
    | generic_renaming_declaration
    | subprogram_renaming_declaration
198 library_unit_body ::= subprogram_body | package_body
199 context_clause ::= {context_item}
200 context_item ::= with_clause | use_clause
201 with_clause ::= limited_with_clause | nonlimited_with_clause
202 limited_with_clause ::=
    limited [private] with library_unit_name {, library_unit_name};
203 nonlimited_with_clause ::=
    [private] with library_unit_name {, library_unit_name};
204 body_stub ::= subprogram_body_stub | package_body_stub
    | task_body_stub | protected_body_stub

```

12 Syntax

```

205 subprogram_body_stub ::=
    [overriding_indicator]
    subprogram_specification is separate
    [aspect_specification];

206 package_body_stub ::= package body defining_identifier is separate
    [aspect_specification];

207 task_body_stub ::= task body defining_identifier is separate
    [aspect_specification];

208 protected_body_stub ::= protected body defining_identifier is separate
    [aspect_specification];

209 subunit ::= separate (parent_unit_name) proper_body

210 renaming_declaration ::=
    object_renaming_declaration
  | exception_renaming_declaration
  | package_renaming_declaration
  | subprogram_renaming_declaration
  | generic_renaming_declaration

211 object_renaming_declaration ::=
    defining_identifier [: [null_exclusion] subtype_mark] renames object_name
    [aspect_specification];
  | defining_identifier : access_definition renames object_name
    [aspect_specification];

212 exception_renaming_declaration ::=
    defining_identifier : exception renames exception_name
    [aspect_specification];

213 package_renaming_declaration ::=
    package defining_program_unit_name renames package_name
    [aspect_specification];

214 subprogram_renaming_declaration ::=
    [overriding_indicator]
    subprogram_specification renames callable_entity_name
    [aspect_specification];

215 generic_renaming_declaration ::=
    generic package defining_program_unit_name
    [aspect_specification];
    renames generic_package_name
  | generic procedure defining_program_unit_name
    [aspect_specification];
    renames generic_procedure_name
  | generic function defining_program_unit_name
    [aspect_specification];
    renames generic_function_name

```

Chapter 14

```

216 record_extension_part ::= with record_definition
217 extension_aggregate ::= (ancestor_part with record_component_association_list)
218 ancestor_part ::= expression | subtype_mark

```

```

219 private_extension_declaration ::=
      type defining_identifier [discriminant_part] is
        [abstract] [limited | synchronized]
        new ancestor_subtype_indication [and interface_list] with private
        [aspect_specification];
220 interface_type_definition ::= [limited | synchronized | task | protected]
      interface [and interface_list]
221 interface_list ::= interface_subtype_mark {and interface_subtype_mark}

```

Chapter 15

```

222 handled_sequence_of_statements ::=
      sequence_of_statements
      [exception
        exception_handler
        {exception_handler}]
223 exception_handler ::= when [choice_parameter_specification :]
      exception_choice { '|' exception_choice } =>
      sequence_of_statements
224 choice_parameter_specification ::= defining_identifier
225 exception_choice ::= exception_name | others
226 exception_declaration ::=
      defining_identifier_list : exception
      [aspect_specification];
227 raise_statement ::= raise; | raise exception_name [with string_expression];
228 raise_expression ::= raise exception_name [with string_simple_expression]

```

Chapter 16

```

229 aspect_specification ::=
      with aspect_mark [ => aspect_definition]
      {, aspect_mark [ => aspect_definition]}
230 aspect_mark ::= aspect_identifier ['Class]
231 aspect_definition ::= name | expression | identifier | aggregate | global_aspect_definition
232 global_aspect_definition ::= null | Unspecified
      | global_mode global_designator
      | (global_aspect_element {; global_aspect_element})
233 global_aspect_element ::= global_mode global_set
      | global_mode all
      | global_mode synchronized
234 global_mode ::= basic_global_mode | extended_global_mode
235 basic_global_mode ::= in | in out | out
236 extended_global_mode ::= overriding basic_global_mode
237 global_set ::= global_name {, global_name}
238 global_designator ::= all | synchronized global_name

```

14 Syntax

239 `global_name ::= object_name | package_name`

Chapter 17

240 `integer_type_definition ::= signed_integer_type_definition | modular_type_definition`

241 `signed_integer_type_definition ::=`
 `range static_simple_expression .. static_simple_expression`

242 `modular_type_definition ::= mod static_expression`

243 `real_type_definition ::= floating_point_definition | fixed_point_definition`

244 `floating_point_definition ::= digits static_expression [real_range_specification]`

245 `real_range_specification ::= range static_simple_expression .. static_simple_expression`

246 `fixed_point_definition ::=`
 `ordinary_fixed_point_definition | decimal_fixed_point_definition`

247 `ordinary_fixed_point_definition ::=`
 `delta static_expression real_range_specification`

248 `decimal_fixed_point_definition ::=`
 `delta static_expression digits static_expression [real_range_specification]`

249 `digits_constraint ::= digits static_simple_expression [range_constraint]`

250 `delta_constraint ::= delta static_simple_expression [range_constraint]`

Chapter 18

251 `discriminant_part ::= unknown_discriminant_part | known_discriminant_part`

252 `unknown_discriminant_part ::= (<>)`

253 `known_discriminant_part ::= (discriminant_specification {; discriminant_specification})`

254 `discriminant_specification ::=`
 `defining_identifier_list : [null_exclusion] subtype_mark [:= default_expression]`
 `[aspect_specification]`
 `| defining_identifier_list : access_definition [:= default_expression]`
 `[aspect_specification]`

255 `discriminant_constraint ::= (discriminant_association {, discriminant_association})`

256 `discriminant_association ::=`
 `[discriminant_selector_name { '|' discriminant_selector_name } =>] expression`

257 `variant_part ::=`
 `case discriminant_direct_name is`
 `variant`
 `{variant}`
 `end case;`

258 `variant ::= when discrete_choice_list => component_list`

Chapter 19

259 `generic_declaration ::= generic_subprogram_declaration | generic_package_declaration`

260 `generic_subprogram_declaration ::=`
 `generic_formal_part subprogram_specification`
 `[aspect_specification];`

```

261 generic_package_declaration ::= generic_formal_part package_specification;
262 generic_formal_part ::= generic {generic_formal_parameter_declaration | use_clause}
263 generic_formal_parameter_declaration ::=
    formal_object_declaration
    | formal_type_declaration
    | formal_subprogram_declaration
    | formal_package_declaration
264 formal_object_declaration ::=
    defining_identifier_list : mode [null_exclusion] subtype_mark
                                [= default_expression]
    [aspect_specification];
    | defining_identifier_list : mode access_definition [= default_expression]
    [aspect_specification];
265 formal_type_declaration ::=
    formal_complete_type_declaration
    | formal_incomplete_type_declaration
266 formal_complete_type_declaration ::=
    type defining_identifier [discriminant_part] is formal_type_definition
    [or use default_subtype_mark] [aspect_specification];
267 formal_incomplete_type_declaration ::=
    type defining_identifier [discriminant_part] [is tagged]
    [or use default_subtype_mark];
268 formal_type_definition ::=
    formal_private_type_definition
    | formal_derived_type_definition
    | formal_discrete_type_definition
    | formal_signed_integer_type_definition
    | formal_modular_type_definition
    | formal_floating_point_definition
    | formal_ordinary_fixed_point_definition
    | formal_decimal_fixed_point_definition
    | formal_array_type_definition
    | formal_access_type_definition
    | formal_interface_type_definition
269 formal_private_type_definition ::=
    [[abstract] tagged] [limited] private
270 formal_derived_type_definition ::=
    [abstract] [limited | synchronized] new subtype_mark
                                [[and interface_list] with private]
271 formal_discrete_type_definition ::= (<>)
272 formal_signed_integer_type_definition ::= range <>
273 formal_modular_type_definition ::= mod <>
274 formal_floating_point_definition ::= digits <>
275 formal_ordinary_fixed_point_definition ::= delta <>
276 formal_decimal_fixed_point_definition ::= delta <> digits <>

```

16 Syntax

```

277 formal_array_type_definition ::= array_type_definition
278 formal_access_type_definition ::= access_type_definition
279 formal_interface_type_definition ::= interface_type_definition
280 formal_subprogram_declaration ::=
    formal_concrete_subprogram_declaration
    | formal_abstract_subprogram_declaration
281 formal_concrete_subprogram_declaration ::=
    with subprogram_specification [is subprogram_default]
    [aspect_specification];
282 formal_abstract_subprogram_declaration ::=
    with subprogram_specification is abstract [subprogram_default]
    [aspect_specification];
283 subprogram_default ::= default_name | <> | null
284 default_name ::= name
285 formal_package_declaration ::=
    with package defining_identifier is
    new generic_package_name formal_package_actual_part
    [aspect_specification];
286 formal_package_actual_part ::=
    ([others =>] <>)
    | [generic_actual_part]
    | (formal_package_association {, formal_package_association} [, others => <>])
287 formal_package_association ::= generic_association
    | generic_formal_parameter_selector_name => <>
288 generic_instantiation ::=
    package defining_program_unit_name is
    new generic_package_name [generic_actual_part]
    [aspect_specification];
    | [overriding_indicator]
    procedure defining_program_unit_name is
    new generic_procedure_name [generic_actual_part]
    [aspect_specification];
    | [overriding_indicator]
    function defining_designator is
    new generic_function_name [generic_actual_part]
    [aspect_specification];
289 generic_actual_part ::= (generic_association {, generic_association})
290 generic_association ::=
    [generic_formal_parameter_selector_name =>]
    explicit_generic_actual_parameter
291 explicit_generic_actual_parameter ::= expression | variable_name
    | subprogram_name | entry_name | subtype_mark
    | package_instance_name

```


Chapter 20

```

292 task_type_declaration ::=
    task type defining_identifier [known_discriminant_part]
    [aspect_specification] is
    [new interface_list with] task_definition;

293 single_task_declaration ::=
    task defining_identifier
    [aspect_specification] is
    [new interface_list with] task_definition;

294 task_definition ::=
    {task_item}
    [private
    {task_item}]
    end [task_identifier]

295 task_item ::= entry_declaration | aspect_clause

296 task_body ::=
    task body defining_identifier
    [aspect_specification] is
    declarative_part
    begin
    handled_sequence_of_statements
    end [task_identifier];

297 entry_declaration ::=
    [overriding_indicator]
    entry defining_identifier [(discrete_subtype_definition)] parameter_profile
    [aspect_specification];

298 entry_call_statement ::= entry_name [actual_parameter_part];

299 accept_statement ::=
    accept entry_direct_name [(entry_index)] parameter_profile [do
    handled_sequence_of_statements
    end [entry_identifier]];

300 entry_index ::= expression

301 delay_statement ::= delay_until_statement | delay_relative_statement

302 delay_until_statement ::= delay until delay_expression;

303 delay_relative_statement ::= delay delay_expression;

304 protected_type_declaration ::=
    protected type defining_identifier [known_discriminant_part]
    [aspect_specification] is
    [new interface_list with] protected_definition;

305 single_protected_declaration ::=
    protected defining_identifier
    [aspect_specification] is
    [new interface_list with] protected_definition;

```

18 Syntax

```

306 protected_definition ::=
    {protected_operation_declaration}
    [private
     {protected_element_declaration}]
    end [protected_identifier]
307 protected_operation_declaration ::= subprogram_declaration
    | entry_declaration | aspect_clause
308 protected_element_declaration ::=
    protected_operation_declaration | component_declaration
309 protected_body ::=
    protected body defining_identifier
    [aspect_specification] is
    {protected_operation_item}
    end [protected_identifier];
310 protected_operation_item ::=
    subprogram_declaration      | subprogram_body
    | null_procedure_declaration | expression_function_declaration
    | entry_body                 | aspect_clause
311 entry_body ::=
    entry defining_identifier entry_body_formal_part
    [aspect_specification]
    entry_barrier is
    declarative_part
    begin
    handled_sequence_of_statements
    end [entry_identifier];
312 entry_body_formal_part ::= [(entry_index_specification)] parameter_profile
313 entry_barrier ::= when condition
314 entry_index_specification ::=
    for defining_identifier in discrete_subtype_definition [aspect_specification]
315 select_statement ::= selective_accept | timed_entry_call
    | conditional_entry_call | asynchronous_select
316 selective_accept ::=
    select
    [guard]
    select_alternative
    {or
     [guard]
     select_alternative}
    [else
     sequence_of_statements]
    end select;
317 guard ::= when condition =>
318 select_alternative ::= accept_alternative | delay_alternative | terminate_alternative
319 accept_alternative ::= accept_statement [sequence_of_statements]
320 delay_alternative ::= delay_statement [sequence_of_statements]

```

```

321 terminate_alternative ::= terminate;
322 conditional_entry_call ::=
    select
        entry_call_alternative
    else
        sequence_of_statements
    end_select;
323 entry_call_alternative ::= procedure_or_entry_call [sequence_of_statements]
324 procedure_or_entry_call ::= procedure_call_statement | entry_call_statement
325 timed_entry_call ::=
    select
        entry_call_alternative
    or
        delay_alternative
    end_select;
326 abort_statement ::= abort task_name {, task_name};
327 asynchronous_select ::=
    select
        triggering_alternative
    then abort
        abortable_part
    end_select;
328 triggering_alternative ::= triggering_statement [sequence_of_statements]
329 triggering_statement ::= procedure_or_entry_call | delay_statement
330 abortable_part ::= sequence_of_statements
331 requeue_statement ::= requeue procedure_or_entry_name [with abort];

```

Chapter 21

```

332 generalized_reference ::= reference_object_name
333 generalized_indexing ::= indexable_container_object_prefix actual_parameter_part

```

Chapter 22

```

334 synchronization_kind ::= By_Entry | By_Protected_Procedure | Optional
335 parallel_block_statement ::=
    parallel [(chunk_specification)] [aspect_specification] do
        sequence_of_statements
    and
        sequence_of_statements
    {and
        sequence_of_statements}
    end do;

```

Chapter 24

```

336 container_aggregate ::= null_container_aggregate | positional_container_aggregate
    | named_container_aggregate

```

20 Syntax

```

337 null_container_aggregate ::= '[' ']'
338 positional_container_aggregate ::= '[' expression {, expression} ']'
339 named_container_aggregate ::= '[' container_element_association_list ']'
340 container_element_association_list ::=
    container_element_association {, container_element_association}
341 container_element_association ::= key_choice_list => expression
    | key_choice_list => <> | iterated_element_association
342 key_choice_list ::= key_choice { '|' key_choice}
343 key_choice ::= key_expression | discrete_range
344 iterated_element_association ::=
    for loop_parameter_specification [use key_expression] => expression
    | for iterator_specification [use key_expression] => expression

```

Chapter 25

```

345 aspect_clause ::=
    attribute_definition_clause | record_representation_clause
    | enumeration_representation_clause | at_clause
346 local_name ::= direct_name | direct_name ' attribute_designator
    | library_unit_name
347 attribute_definition_clause ::=
    for local_name ' attribute_designator use expression;
    | for local_name ' attribute_designator use name;
348 enumeration_representation_clause ::=
    for first_subtype_local_name use enumeration_aggregate;
349 enumeration_aggregate ::= array_aggregate
350 record_representation_clause ::=
    for first_subtype_local_name use
    record [mod_clause]
    {component_clause}
    end record [local_name];
351 mod_clause ::= at mod static_expression;
352 component_clause ::= component_local_name at position range first_bit .. last_bit;
353 position ::= static_expression
354 first_bit ::= static_simple_expression
355 last_bit ::= static_simple_expression
356 at_clause ::= for direct_name use at expression;
357 code_statement ::= qualified_expression;
358 storage_pool_indicator ::= storage_pool_name | null | Standard

```

Chapter 26

```

359 formal_parameter_set ::= formal_group_designator | formal_parameter_name
    | (formal_parameter_name {, formal_parameter_name})

```

360 formal_group_designator ::= **null** | **all**
 361 formal_parameter_name ::= *formal_subtype_mark* | *formal_subprogram_name* |
 formal_access_to_subprogram_object_name
 362 dispatching_operation_set ::= dispatching_operation_specifier |
 (dispatching_operation_specifier {, dispatching_operation_specifier}
 363 dispatching_operation_specifier ::= *dispatching_operation_name* (*object_name*)

Note that the syntactic terms *at_clause*, *delta_constraint*, and *mod_clause* are obsolescent.

2 Syntax index

This index lists the syntactic categories in alphabetical order and gives the number of their definition in the previous section and the numbers of those in which each is used. Categories with a definition number of zero are terminal.

Category	Definition number	Used in number					
abort_statement	326	67					
abortable_part	330	327					
abstract_subprogram_declaration	190	19					
accept_alternative	319	318					
accept_statement	299	68	319				
access_definition	173	20	85	99	147	155	166
		211	254	264			
access_to_object_definition	169	168					
access_to_subprogram_definition	171	168					
access_type_definition	168	30	278				
actual_parameter_part	160	158	159	298	333		
aggregate	104	56	64	167	231		
allocator	175	56					
ancestor_part	218	217					
array_aggregate	109	104	349				
array_component_association	114	113					
array_component_association_list	113	112	118				
array_delta_aggregate	118	116					
array_type_definition	94	20	30	277			
aspect_clause	345	122	182	295	307	310	
aspect_definition	231	229					
aspect_mark	230	18	229				
aspect_specification	229	20	29	31	80	123	139
		142	155	157	164	167	178
		179	188	190	191	205	206
		207	208	211	212	213	214
		215	219	226	254	260	264
		266	281	282	285	288	292
		293	296	297	304	305	309
		311	314	335			

22 Syntax

Category	Definition number	Used in number					
assignment_statement	24	67					
asynchronous_select	327	315					
at_clause	356	345					
attribute_definition_clause	347	345					
attribute_designator	49	47	346	347			
attribute_reference	47	42					
base	10	9					
based_literal	9	4					
based_numeral	11	9					
basic_declaration	19	182					
basic_declarative_item	182	178	181				
basic_global_mode	235	234	236				
binary_adding_operator	59	53					
block_statement	26	68					
body	183	181					
body_stub	204	183					
case_expression	131	129					
case_expression_alternative	132	131					
case_statement	73	68					
case_statement_alternative	74	73					
character	0	16					
character_literal	13	41	42	126			
choice_expression	77	76					
choice_parameter_specification	224	223					
choice_relation	78	77					
chunk_specification	81	80	139	335			
code_statement	357	67					
comment	16	not used					
compilation	193	not used					
compilation_unit	194	193					
component_choice_list	108	107					
component_clause	352	350					
component_declaration	123	122	308				
component_definition	99	95	96	123			
component_item	122	121					
component_list	121	120	258				
composite_constraint	36	34					
compound_statement	68	66					
condition	72	71	80	83	92	130	313
		317					
conditional_entry_call	322	315					
conditional_expression	129	56					
constrained_array_definition	96	94					
constraint	34	32					
container_aggregate	336	104					

Category	Definition number	Used in number					
container_element_association	341	340					
container_element_association_list	340	339					
context_clause	199	194					
context_item	200	199					
decimal_fixed_point_definition	248	246					
decimal_literal	5	4					
declarative_item	181	180					
declarative_part	180	26	157	179	296	311	
declare_expression	136	56					
declare_item	137	136					
default_expression	124	123	155	254	264		
default_name	284	283					
defining_character_literal	41	40					
defining_designator	149	145	288				
defining_identifier	23	22	29	31	40	81	82
		84	87	115	150	164	174
		188	206	207	208	211	212
		219	224	266	267	285	292
		293	296	297	304	305	309
		311	314				
defining_identifier_list	22	20	21	123	155	226	254
		264					
defining_operator_symbol	153	149					
defining_program_unit_name	150	144	149	178	179	213	215
		288					
delay_alternative	320	318	325				
delay_relative_statement	303	301					
delay_statement	301	67	320	329			
delay_until_statement	302	301					
delta_aggregate	116	104					
delta_constraint	250	35					
derived_type_definition	189	30					
designator	148	157					
digit	8	6	12				
digits_constraint	249	35					
direct_name	43	27	42	257	299	346	356
discrete_choice	76	75					
discrete_choice_list	75	74	114	115	132	258	
discrete_range	101	100	103	343			
discrete_subtype_definition	98	81	82	96	297	314	
discriminant_association	256	255					
discriminant_constraint	255	36					
discriminant_part	251	174	188	219	266	267	
discriminant_specification	254	253					
dispatching_operation_set	362	not used					

24 Syntax

Category	Definition number	Used in number						
dispatching_operation_specifier	363	362						
entry_barrier	313	311						
entry_body	311	310						
entry_body_formal_part	312	311						
entry_call_alternative	323	322	325					
entry_call_statement	298	67	324					
entry_declaration	297	295	307					
entry_index	300	299						
entry_index_specification	314	312						
enumeration_aggregate	349	348						
enumeration_literal_specification	40	39						
enumeration_representation_clause	348	345						
enumeration_type_definition	39	30						
exception_choice	225	223						
exception_declaration	226	19						
exception_handler	223	222						
exception_renaming_declaration	212	210						
exit_statement	92	67						
explicit_actual_parameter	162	161						
explicit_dereference	45	42						
explicit_generic_actual_parameter	291	290						
exponent	7	5	9					
expression	51	18	20	21	24	49	50	
		56	63	64	72	73	102	
		107	110	114	115	117	118	
		124	130	131	132	135	136	
		162	163	164	167	218	227	
		231	242	244	247	248	256	
		291	300	302	303	338	341	
		343	344	347	351	352	356	
		167	19	310				
		12	11					
expression_function_declaration	167	19	310					
extended_digit	12	11						
extended_global_mode	236	234						
extended_return_object_declaration	164	165						
extended_return_statement	165	68						
extension_aggregate	217	104						
factor	55	54						
first_bit	354	352						
fixed_point_definition	246	243						
floating_point_definition	244	243						
formal_abstract_subprogram_declaration	282	280						
formal_access_type_definition	278	268						
formal_array_type_definition	277	268						
formal_complete_type_declaration	266	265						
formal concrete subprogram declaration	281	280						

Category	Definition number	Used in number		
formal_decimal_fixed_point_definition	276	268		
formal_derived_type_definition	270	268		
formal_discrete_type_definition	271	268		
formal_floating_point_definition	274	268		
formal_group_designator	360	359		
formal_incomplete_type_declaration	267	265		
formal_interface_type_definition	279	268		
formal_modular_type_definition	273	268		
formal_object_declaration	264	263		
formal_ordinary_fixed_point_definition	275	268		
formal_package_actual_part	286	285		
formal_package_association	287	286		
formal_package_declaration	285	263		
formal_parameter_name	361	359		
formal_parameter_set	359	not used		
formal_part	154	87	146	147
formal_private_type_definition	269	268		
formal_signed_integer_type_definition	272	268		
formal_subprogram_declaration	280	263		
formal_type_declaration	265	263		
formal_type_definition	268	266		
full_type_declaration	29	28		
function_call	159	42		
function_specification	145	143	167	
general_access_modifier	170	169		
generalized_indexing	333	42		
generalized_reference	332	42		
generic_actual_part	289	286	288	
generic_association	290	287	289	
generic_declaration	259	19	196	
generic_formal_parameter_declaration	263	262		
generic_formal_part	262	260	261	
generic_instantiation	288	19	196	
generic_package_declaration	261	259		
generic_renaming_declaration	215	197	210	
generic_subprogram_declaration	260	259		
global_aspect_definition	232	231		
global_aspect_element	233	232		
global_designator	238	232		
global_mode	234	232	233	
global_name	239	237	238	
global_set	237	233		
goto_statement	93	67		
graphic_character	0	13	15	
guard	317	316		

26 Syntax

Category	Definition number	Used in number					
handled_sequence_of_statements	222	26 311	157	165	179	296	299
highest_precedence_operator	62	not used					
identifier	1	17 79 230 309	18 120 231 311	23 126 294	26 148 296	43 178 299	49 179 306
identifier_extend	3	1					
identifier_start	2	1					
if_expression	130	129					
if_statement	71	68					
implicit_dereference	46	44					
incomplete_type_declaration	174	28					
index_constraint	100	36					
index_subtype_definition	97	95					
indexed_component	102	42					
integer_type_definition	240	30					
interface_list	221	189 304	219 305	220	270	292	293
interface_type_definition	220	30	279				
iterated_component_association	115	114					
iterated_element_association	344	139	341				
iteration_scheme	80	79					
iterator_actual_parameter_part	89	88					
iterator_filter	83	82	84	86			
iterator_parameter_association	90	89					
iterator_parameter_specification	87	86					
iterator_procedure_call	88	86					
iterator_specification	84	80	115	133			
key_choice	343	342					
key_choice_list	342	341					
known_discriminant_part	253	29	251	292	304		
label	69	65	66				
last_bit	355	352					
letter_lowercase	0	2					
letter_modifier	0	2					
letter_other	0	2					
letter_titlecase	0	2					
letter_uppercase	0	2					
library_item	195	194					
library_unit_body	198	195					
library_unit_declaration	196	195					
library_unit_renaming_declaration	197	195					
limited_with_clause	202	201					

Category	Definition number	Used in number					
local_name	346	347	348	350	352		
logical_operator	57	not used					
loop_parameter_specification	82	80	133	344			
loop_parameter_subtype_indication	85	84					
loop_statement	79	68					
mark_nonspacing	0	3					
mark_spacing_combining	0	3					
membership_choice	128	127					
membership_choice_list	127	52					
mod_clause	351	350					
mode	156	155	264				
modular_type_definition	242	240					
multiplying_operator	61	54					
name	42	18	24	33	44	45	46
		56	63	84	88	92	93
		151	158	159	162	176	186
		202	203	211	212	213	214
		215	225	227	228	231	239
		284	285	288	291	298	326
		331	332	346	347	358	361
		363					
name_expression	0	141					
named_array_aggregate	112	109					
named_container_aggregate	339	336					
nonlimited_with_clause	203	201					
null_array_aggregate	111	109					
null_container_aggregate	337	336					
null_exclusion	172	32	147	155	168	173	211
		254	264				
null_procedure_declaration	191	19	310				
null_statement	70	67					
number_decimal	0	3					
number_declaration	21	19					
number_letter	0	2					
numeral	6	5	7	10			
numeric_literal	4	56					
object_declaration	20	19	137				
object_renaming_declaration	211	137	210				
operator_symbol	152	43	126	148	153		
ordinary_fixed_point_definition	247	246					
overriding_indicator	192	142	157	167	190	191	205
		214	288	297			
package_body	179	184	198				
package_body_stub	206	204					

28 Syntax

Category	Definition number	Used in number				
package_declaration	177	19	196			
package_renaming_declaration	213	197	210			
package_specification	178	177	261			
parallel_block_statement	335	68				
parameter_and_result_profile	147	145	171	173		
parameter_association	161	90	160			
parameter_association_with_box	91	90				
parameter_profile	146	144	171	173	297	299 312
parameter_specification	155	154				
parent_unit_name	151	148	150	178	179	209
position	353	352				
positional_array_aggregate	110	109				
positional_container_aggregate	338	336				
pragma	17	not used				
pragma_argument_association	18	17				
predicate	135	133				
prefix	44	47	48	88	102	103 125
		138	158	159	333	
primary	56	55				
private_extension_declaration	219	28				
private_type_declaration	188	28				
procedural_iterator	86	80				
procedure_call_statement	158	67	324			
procedure_or_entry_call	324	323	329			
procedure_specification	144	143	191			
proper_body	184	183	209			
protected_body	309	184				
protected_body_stub	208	204				
protected_definition	306	304	305			
protected_element_declaration	308	306				
protected_operation_declaration	307	306	308			
protected_operation_item	310	309				
protected_type_declaration	304	29				
punctuation_connector	0	3				
qualified_expression	64	42	175	356		
quantified_expression	133	56				
quantifier	134	133				
raise_expression	228	52				
raise_statement	227	67				
range	38	37	76	98	101	128
range_attribute_designator	50	48				
range_attribute_reference	48	38				
range_constraint	37	35	249	250		
real_range_specification	245	244	247	248		
real_type_definition	243	30				

Category	Definition number	Used in number						
record_aggregate	105	104						
record_component_association	107	106						
record_component_association_list	106	105	117	217				
record_definition	120	119	216					
record_delta_aggregate	117	116						
record_extension_part	216	189						
record_representation_clause	350	345						
record_type_definition	119	30						
reduction_attribute_designator	140	138						
reduction_attribute_reference	138	47						
reduction_identifier	0	140						
reduction_specification	141	140						
relation	52	51						
relational_operator	58	52	78					
renaming_declaration	210	19						
requeue_statement	331	67						
return_subtype_indication	166	164						
scalar_constraint	35	34						
select_alternative	318	316						
select_statement	315	68						
selected_component	125	42						
selective_accept	316	315						
selector_name	126	91	108	125	161	256	287	
		290						
sequence_of_statements	65	71	74	79	222	223	316	
		319	320	322	323	328	330	
		335						
signed_integer_type_definition	241	240						
simple_expression	53	38	52	78	81	128	228	
		241	245	249	250	353	354	
simple_return_statement	163	67						
simple_statement	67	66						
single_protected_declaration	305	20						
single_task_declaration	293	20						
slice	103	42						
statement	66	65						
statement_identifier	27	26	69	79				
storage_pool_indicator	358	not used						
string_element	15	14						
string_literal	14	56	152					
subpool_specification	176	175						
subprogram_body	157	184	198	310				
subprogram_body_stub	205	204						
subprogram_declaration	142	19	196	307	310			
subprogram_default	283	281	282					

30 Syntax

Category	Definition number	Used in number					
subprogram_renaming_declaration	214	197	210				
subprogram_specification	143	142	157	190	205	214	260
		281	282				
subtype_declaration	31	19					
subtype_indication	32	20	31	76	85	98	99
		101	166	169	175	189	219
subtype_mark	33	32	63	64	97	128	147
		155	173	187	211	218	221
		254	264	266	267	270	291
		361					
subunit	209	194					
synchronization_kind	334	not used					
target_name	25	42					
task_body	296	184					
task_body_stub	207	204					
task_definition	294	292	293				
task_item	295	294					
task_type_declaration	292	29					
term	54	53					
terminate_alternative	321	318					
timed_entry_call	325	315					
triggering_alternative	328	327					
triggering_statement	329	328					
type_conversion	63	42					
type_declaration	28	19					
type_definition	30	29					
unary_adding_operator	60	53					
unconstrained_array_definition	95	94					
underline	0	6	11				
unknown_discriminant_part	252	251					
use_clause	185	182	200	262			
use_package_clause	186	185					
use_type_clause	187	185					
value_sequence	139	138					
variant	258	257					
variant_part	257	121					
with_clause	201	200					