Ada Programming/All Keywords

Aus Wikibooks
< Ada Programming

Contents

1 Keywords
  1.1 Language summary keywords
  1.2 List of keywords
  1.3 See also
    1.3.1 Wikibook
    1.3.2 Ada 95 Reference Manual
    1.3.3 Ada 2005 Reference Manual
    1.3.4 Ada Quality and Style Guide

2 Keywords: abort
  2.1 abort
  2.2 See also
    2.2.1 Wikibook
    2.2.2 Ada Reference Manual
    2.2.3 Ada Quality and Style Guide

3 Keywords: abs
  3.1 See also
    3.1.1 Wikibook
    3.1.2 Ada 95 Reference Manual
    3.1.4 Ada Quality and Style Guide

4 Keywords: abstract
  4.1 Summary
  4.2 See also
    4.2.1 Wikibook
    4.2.2 Ada Reference Manual
    4.2.3 Ada Quality and Style Guide

5 Keywords: accept
  5.1 Summary
  5.2 See also
    5.2.1 Wikibook
    5.2.2 Ada Reference Manual
    5.2.3 Ada Quality and Style Guide

6 Keywords: access
  6.1 See also
    6.1.1 Wikibook
    6.1.2 Ada Reference Manual
    6.1.3 Ada Quality and Style Guide

7 Keywords: aliased
  7.1 Description
7.2 For variables
7.3 For type declarations
  7.3.1 For the elements of an array
  7.3.2 For the elements of a record
7.4 See also
  7.4.1 Wikibook
  7.4.2 Ada Reference Manual
  7.4.3 Ada Quality and Style Guide
8 Keywords: all
  8.1 See also
    8.1.1 Wikibook
    8.1.2 Ada Reference Manual
    8.1.3 Ada Quality and Style Guide
9 Keywords: and
  9.1 Logical operator
    9.1.1 Boolean operator
    9.1.2 Boolean shortcut operator
    9.1.3 Boolean operator on arrays
    9.1.4 Bitwise operator
  9.2 Adding interfaces to tagged types
  9.3 See also
    9.3.1 Wikibook
    9.3.2 Ada 95 Reference Manual
    9.3.2.1 Ada 2005 Reference Manual
    9.3.3 Ada Quality and Style Guide
10 Keywords: array
  10.1 See also
    10.1.1 Wikibook
    10.1.2 Ada Reference Manual
    10.1.3 Ada Quality and Style Guide
11 Keywords: at
  11.1 See also
    11.1.1 Wikibook
    11.1.2 Ada Reference Manual
    11.1.3 Ada Quality and Style Guide
12 Keywords: begin
  12.1 See also
    12.1.1 Wikibook
    12.1.2 Ada Reference Manual
    12.1.3 Ada Quality and Style Guide
13 Keywords: body
  13.1 See also
    13.1.1 Wikibook
    13.1.3 Ada Quality and Style Guide
14 Keywords: case
  14.1 See also
    14.1.1 Wikibook
    14.1.3 Ada Quality and Style Guide
15 Keywords: constant
  15.1 See also
    15.1.1 Wikibook
    15.1.2 Ada Reference Manual
    15.1.3 Ada Quality and Style Guide
16 Keywords: declare
  16.1 See also
    16.1.1 Wikibook
    16.1.2 Ada 95 Reference Manual
16.1.4 Ada Quality and Style Guide

17 Keywords: delay
   17.1 Delay statements
      17.1.1 Relative delays
      17.1.2 Absolute delays
   17.2 Select statements
      17.2.1 Delay alternative
      17.2.2 Timed entry call
      17.2.3 Asynchronous transfer of control
   17.3 See also
      17.3.1 Wikibook
      17.3.2 Ada Reference Manual
      17.3.3 Ada Quality and Style Guide

18 Keywords: delta
   18.1 Description
   18.2 Example
   18.3 See also
      18.3.1 Wikibook
      18.3.2 Ada Reference Manual
      18.3.3 Ada Quality and Style Guide

19 Keywords: digits
   19.1 See also
      19.1.1 Wikibook
      19.1.3 Ada Quality and Style Guide

20 Keywords: do
   20.1 See also
      20.1.1 Wikibook
      20.1.2 Ada Reference Manual
      20.1.3 Ada Quality and Style Guide

21 Keywords: else
   21.1 See also
      21.1.1 Wikibook
      21.1.3 Ada Quality and Style Guide

22 Keywords: elsif
   22.1 See also
      22.1.1 Wikibook
      22.1.2 Ada Reference Manual
      22.1.3 Ada Quality and Style Guide

23 Keywords: end
   23.1 See also
      23.1.1 Wikibook
      23.1.2 Ada Reference Manual
      23.1.3 Ada Quality and Style Guide

24 Keywords: entry
   24.1 See also
      24.1.1 Wikibook
      24.1.2 Ada Reference Manual
      24.1.3 Ada Quality and Style Guide

25 Keywords: exception
   25.1 See also
      25.1.1 Wikibook
      25.1.2 Ada Reference Manual
      25.1.3 Ada Quality and Style Guide

26 Keywords: exit
   26.1 See also
      26.1.1 Wikibook
27 Keywords: for
   27.1 See also
   27.1.1 Wikibook
   27.1.2 Ada Reference Manual
   27.1.3 Ada Quality and Style Guide

28 Keywords: function
   28.1 See also
   28.1.1 Wikibook
   28.1.3 Ada Quality and Style Guide

29 Keywords: generic
   29.1 See also
   29.1.1 Wikibook
   29.1.2 Ada Reference Manual
   29.1.3 Ada Quality and Style Guide

30 Keywords: goto
   30.1 See also
   30.1.1 Wikibook
   30.1.2 Ada Reference Manual
   30.1.3 Ada Quality and Style Guide

31 Keywords: if
   31.1 See also
   31.1.1 Wikibook
   31.1.2 Ada Reference Manual
   31.1.3 Ada Quality and Style Guide

32 Keywords: in
   32.1 See also
   32.1.1 Wikibook
   32.1.2 Ada 95 Reference Manual
   32.1.4 Ada Quality and Style Guide

33 Keywords: interface
   33.1 Interface types
   33.1.1 Normal interface
   33.1.2 Limited interface
   33.1.3 Synchronized interface
   33.1.4 Task interface
   33.1.5 Protected interface
   33.2 See also
   33.2.1 Wikibook
   33.2.2 Ada Reference Manual
   33.2.2.1 Ada 2005
   33.2.3 Ada Quality and Style Guide

34 Keywords: is
   34.1 In type declarations
   34.2 In subtype declarations
   34.3 In package declarations
   34.4 In procedure and function declaration
   34.5 In generic instantiations
   34.6 See also
   34.6.1 Wikibook
   34.6.2 Ada Reference Manual
   34.6.3 Ada Quality and Style Guide

35 Keywords: limited
   35.1 See also
   35.1.1 Wikibook
   35.1.2 Ada Reference Manual
35.1.3 Ada Quality and Style Guide

36 Keywords: loop
   36.1 See also
      36.1.1 Wikibook
      36.1.2 Ada Reference Manual
      36.1.3 Ada Quality and Style Guide

37 Keywords: mod
   37.1 See also
      37.1.1 Wikibook
      37.1.2 Ada 95 Reference Manual
      37.1.4 Ada Quality and Style Guide

38 Keywords: new
   38.1 Declaration
      38.1.1 In type derivation
      38.1.2 In type extension
      38.1.3 In instantiation of generics
      38.2 Allocator
      38.3 See also
         38.3.1 Wikibook
         38.3.2 Ada Reference Manual
         38.3.3 Ada Quality and Style Guide

39 Keywords: not
   39.1 See also
      39.1.1 Wikibook
      39.1.2 Ada 95 Reference Manual
      39.1.4 Ada Quality and Style Guide

40 Keywords: null
   40.1 See also
      40.1.1 Wikibook
      40.1.2 Ada Reference Manual
      40.1.3 Ada Quality and Style Guide

41 Keywords: of
   41.1 See also
      41.1.1 Wikibook
      41.1.2 Ada Reference Manual
      41.1.3 Ada Quality and Style Guide

42 Keywords: or
   42.1 Logical operator
      42.1.1 Boolean operator
      42.1.2 Boolean shortcut operator
      42.1.3 Boolean operator on arrays
      42.1.4 Bitwise operator
   42.2 Select statement
      42.2.1 alternative
      42.2.2 delay
      42.3 See also
         42.3.1 Wikibook
         42.3.2 Ada 95 Reference Manual
         42.3.3 Ada 2005 Reference Manual
         42.3.4 Ada Quality and Style Guide

43 Keywords: others
   43.1 Summary
   43.2 See also
      43.2.1 Wikibook
      43.2.2 Ada Reference Manual
      43.2.3 Ada Quality and Style Guide

44 Keywords: out
44.1 See also
   - 44.1.1 Wikibook
   - 44.1.2 Ada 95 Reference Manual
   - 44.1.3 Ada 2005 Reference Manual
   - 44.1.4 Ada Quality and Style Guide

45 Keywords: overriding
   - 45.1 See also
     - 45.1.1 Wikibook
     - 45.1.2 Ada Reference Manual
     - 45.1.3 Ada Quality and Style Guide

46 Keywords: package
   - 46.1 See also
     - 46.1.1 Wikibook
     - 46.1.2 Ada 95 Reference Manual
     - 46.1.4 Ada Quality and Style Guide

47 Keywords: pragma
   - 47.1 See also
     - 47.1.1 Wikibook
     - 47.1.2 Ada Reference Manual
     - 47.1.3 Ada Quality and Style Guide

48 Keywords: private
   - 48.1 See also
     - 48.1.1 Wikibook
     - 48.1.3 Ada Quality and Style Guide

49 Keywords: procedure
   - 49.1 See also
     - 49.1.1 Wikibook
     - 49.1.2 Ada Reference Manual
     - 49.1.3 Ada Quality and Style Guide

50 Keywords: protected
   - 50.1 See also
     - 50.1.1 Wikibook
     - 50.1.3 Ada Quality and Style Guide

51 Keywords: raise
   - 51.1 See also
     - 51.1.1 Wikibook
     - 51.1.2 Ada Reference Manual
     - 51.1.3 Ada Quality and Style Guide

52 Keywords: range
   - 52.1 Description
   - 52.2 See also
     - 52.2.1 Wikibook
     - 52.2.2 Ada Reference Manual
     - 52.2.3 Ada Quality and Style Guide

53 Keywords: record
   - 53.1 See also
     - 53.1.1 Wikibook
     - 53.1.3 Ada Quality and Style Guide

54 Keywords: rem
   - 54.1 Operator rem
   - 54.2 See also
     - 54.2.1 Wikibook
     - 54.2.2 Ada 95 Reference Manual
     - 54.2.3 Ada 2005 Reference Manual
     - 54.2.4 Ada Quality and Style Guide
55 Keywords: renames
   55.1 See also
      55.1.1 Wikibook
      55.1.2 Ada Reference Manual
      55.1.3 Ada Quality and Style Guide

56 Keywords: requeue
   56.1 See also
      56.1.1 Wikibook
      56.1.2 Ada Reference Manual
      56.1.3 Ada Quality and Style Guide

57 Keywords: return
   57.1 See also
      57.1.1 Wikibook
      57.1.2 Ada 95 Reference Manual
      57.1.4 Ada Quality and Style Guide

58 Keywords: reverse
   58.1 See also
      58.1.1 Wikibook
      58.1.2 Ada Reference Manual
      58.1.3 Ada Quality and Style Guide

59 Keywords: select
   59.1 See also
      59.1.1 Wikibook
      59.1.2 Ada Reference Manual
      59.1.3 Ada Quality and Style Guide

60 Keywords: separate
   60.1 See also
      60.1.1 Wikibook
      60.1.2 Ada Reference Manual
      60.1.3 Ada Quality and Style Guide

61 Keywords: subtype
   61.1 Summary
   61.2 See also
      61.2.1 Wikibook
      61.2.2 Ada Reference Manual
      61.2.3 Ada Quality and Style Guide

62 Keywords: synchronized
   62.1 See also
      62.1.1 Wikibook
      62.1.2 Ada Reference Manual
      62.1.3 Ada Quality and Style Guide

63 Keywords: tagged
   63.1 See also
      63.1.1 Wikibook
      63.1.2 Ada Reference Manual
      63.1.3 Ada Quality and Style Guide

64 Keywords: task
   64.1 See also
      64.1.1 Wikibook
      64.1.2 Ada Reference Manual
      64.1.2.1 Ada 95
      64.1.2.2 Ada 2005
      64.1.3 Ada Quality and Style Guide

65 Keywords: terminate
   65.1 See also
      65.1.1 Wikibook
      65.1.2 Ada Reference Manual
      65.1.3 Ada Quality and Style Guide
1 Keywords

- 66 Keywords: then
  - 66.1 See also
    - 66.1.1 Wikibook
    - 66.1.3 Ada Quality and Style Guide

- 67 Keywords: type
  - 67.1 See also
    - 67.1.1 Wikibook
    - 67.1.3 Ada Quality and Style Guide

- 68 Keywords: until
  - 68.1 Delay until statement
  - 68.2 See also
    - 68.2.1 Wikibook
    - 68.2.2 Ada Reference Manual
    - 68.2.3 Ada Quality and Style Guide

- 69 Keywords: use
  - 69.1 Use clause
    - 69.1.1 At library level
    - 69.1.2 At declaration level
    - 69.1.3 For types
  - 69.2 Representation clause
    - 69.2.1 For attributes
    - 69.2.2 For records
    - 69.2.3 For enumerations
  - 69.3 See also
    - 69.3.1 Wikibook
    - 69.3.2 Ada Reference Manual
    - 69.3.3 Ada Quality and Style Guide

- 70 Keywords: when
  - 70.1 See also
    - 70.1.1 Wikibook
    - 70.1.2 Ada Reference Manual
    - 70.1.3 Ada Quality and Style Guide

- 71 Keywords: while
  - 71.1 See also
    - 71.1.1 Wikibook
    - 71.1.2 Ada Reference Manual
    - 71.1.3 Ada Quality and Style Guide

- 72 Keywords: with
  - 72.1 See also
    - 72.1.1 Wikibook
    - 72.1.2 Ada Reference Manual
    - 72.1.3 Ada Quality and Style Guide

- 73 Keywords: xor
  - 73.1 Logical operator
    - 73.1.1 Boolean operator
    - 73.1.2 Boolean operator on arrays
    - 73.1.3 Bitwise operator
  - 73.2 See also
    - 73.2.1 Wikibook
    - 73.2.2 Ada 95 Reference Manual
    - 73.2.3 Ada 2005 Reference Manual
    - 73.2.4 Ada Quality and Style Guide
1.1 Language summary keywords

Most Ada **keywords** have different functions depending on where they are used. A good example is **for** which controls the representation clause when used within a declaration part and controls a loop when used within an implementation.

In Ada, a keyword is also a reserved word, so it cannot be used as an identifier.

1.2 List of keywords

<table>
<thead>
<tr>
<th>Ada Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>abort</td>
</tr>
<tr>
<td>abs</td>
</tr>
<tr>
<td>abstract</td>
</tr>
<tr>
<td>accept</td>
</tr>
<tr>
<td>access</td>
</tr>
<tr>
<td>aliased</td>
</tr>
<tr>
<td>all</td>
</tr>
<tr>
<td>and</td>
</tr>
<tr>
<td>at</td>
</tr>
<tr>
<td>begin</td>
</tr>
<tr>
<td>body</td>
</tr>
<tr>
<td>case</td>
</tr>
<tr>
<td>constant</td>
</tr>
<tr>
<td>declare</td>
</tr>
<tr>
<td>delay</td>
</tr>
<tr>
<td>delta</td>
</tr>
<tr>
<td>digits</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

1.3 See also

1.3.1 Wikibook

- Ada Programming

1.3.2 Ada 95 Reference Manual


1.3.3 Ada 2005 Reference Manual

2 Keywords: abort

2.1 abort

The abort is used to abort either a task (thread) or partition (process).

2.2 See also

2.2.1 Wikibook

- Ada Programming
- Ada Programming/Keywords
- Ada Programming/Tasking

2.2.2 Ada Reference Manual

- 2.9 Reserved Words
- Annex P Syntax Summary
- 9.8 Abort of a Task
- E.1 Partitions

2.2.3 Ada Quality and Style Guide

- 3.1.3 Capitalization

3 Keywords: abs

This keyword is used for the operator that gets the absolute value of an integer number.

\[ y := \text{abs}(x); \]
3.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

3.1.2 Ada 95 Reference Manual


3.1.4 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)
- 5.5.3 Parenthetical Expressions (http://www.adaic.org/docs/95style/html/sec_5/5-5-3.html)

4 Keywords: abstract

4.1 Summary

The keyword `abstract` is used to define an abstract tagged type. See Ada Programming/Object Orientation for details on object orientation in Ada.

4.2 See also

4.2.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

4.2.2 Ada Reference Manual

5 Keywords: accept

5.1 Summary

The keyword accept is used in Ada tasks for accepting a rendezvous.

5.2 See also

5.2.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

5.2.2 Ada Reference Manual


5.2.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adacii.org/docs/95style/html/sec_3/3-1-3.html)

6 Keywords: access

This keyword is used in access types declarations and anonymous access parameters.

6.1 See also

6.1.1 Wikibook
7 Keywords: aliased

7.1 Description

If you come from C/C++ you are probably used to the fact that every element of an array, record and other variables has an address. The C/C++ standards actually demands that. In Ada this is not true.

Ada is a self optimizing language - there is for example no register keyword like in C/C++. Ada compilers will use a register for storage automatically. Incidentally, most C/C++ compilers nowadays just ignore the register and allocate registers themself, just like Ada does.

So if you want to take an access from any variable you need to tell the compiler that the variable needs to be in memory and may not reside inside a register. This is what the keyword aliased is for.

7.2 For variables

```ada
I : aliased Integer := 0;
```

7.3 For type declarations

7.3.1 For the elements of an array

Declaring an array as aliased will only ensure that the array as a whole has an address. It says nothing about the individual elements of the array - which may be packed in a way that more then one element has the same address. You need to declare the actual elements as aliased as well. You can read in Types:array how this is done. Here is just a short example:

```ada
type Day_Of_Month is range 1 .. 31; type Day_Has_Appointment is array (Day_Of_Month) of aliased Boolean;
```

7.3.2 For the elements of a record
Just like arrays, declaring a record as aliased will only ensure that the record as a whole has an address. It says nothing about the individual elements the record -- which again may be packed and share addresses. Again you need to declare the actual elements as aliased as well. You can read in Types:record how this is done. Here is just a short example:

```ada
type Basic_Record is
  record
    A : aliased Integer;
  end record;
end
```

### 7.4 See also

#### 7.4.1 Wikibook
- Ada Programming
- Ada Programming/Keywords

#### 7.4.2 Ada Reference Manual

#### 7.4.3 Ada Quality and Style Guide
- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

### 8 Keywords: all

This keyword is used in:
- General access types declaration
- Explicit dereferencing of access variables

### 8.1 See also

#### 8.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords

#### 8.1.2 Ada Reference Manual
8.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

9 Keywords: and

9.1 Logical operator

9.1.1 Boolean operator

\[
\begin{align*}
X : \text{Boolean} &:= A > 10 \quad \text{and} \quad A < 20; \\
\end{align*}
\]

9.1.2 Boolean shortcut operator

Shortcut operators are used to make the evaluation of parts of boolean expressions conditional: and then, or else. This should never be done to speed up the evaluation (with modern optimizing compilers, it will possibly not have that effect). The correct use is to prevent the evaluation of expressions known to raise an exception.

\[
\begin{align*}
\text{if } \text{Dog} \neq \text{null} \quad \text{and then} \quad \text{G (Dog)} \text{ then} \\
&\quad \text{Walk (Dog);} \\
&\quad \text{end if;}
\end{align*}
\]

In the example above, G (Dog) is only called when the pointer Dog is not null, i.e. it actually points to something.

Actually and then and or else are not operators in the sense of the reference manual, they are called 'Short-circuit Control Forms'. The difference is that (true) operators can be redefined (i.e. overloaded), whereas these cannot. They are however defined for any boolean type.

Since Ada allows parallel evaluation of the arguments for an expression, shortcut operators are not the standard way of evaluating boolean expressions. In case the final result of the evaluation is guaranteed to be the same, the compiler is allowed to use a shortcut evaluation.

9.1.3 Boolean operator on arrays

The and operator is applied to each pair of boolean elements from the left and right arrays. The result has the same bounds than the left operand.

\[
\begin{align*}
\text{type} \, \text{Day} \_\text{Of} \_\text{Month} & \text{ is range } 1 .. 31; \\
\text{type} \, \text{Month} \_\text{Array} & \text{ is array } (\text{Day} \_\text{Of} \_\text{Month}) \text{ of Boolean;} \\
\text{X} : \text{Month} \_\text{Array} &:= \text{Function} \_1; \\
\text{Y} : \text{Month} \_\text{Array} &:= \text{Function} \_2; \\
\text{Z} : \text{Month} \_\text{Array} &:= \text{X} \quad \text{and} \quad \text{Y};
\end{align*}
\]

9.1.4 Bitwise operator

The operator and could be used with modular types to perform bitwise operations.
9.2 Adding interfaces to tagged types

This language feature will be made available in the forthcoming Ada 2005 standard.

```ada
type Programmer is new Person
and Printable
with
record
  Skilled_In : Language_List;
end record;
```

9.3 See also

9.3.1 Wikibook
- Ada Programming
- Ada Programming/Keywords
- Ada Programming/Operators
- Ada Programming/Keywords/interface

9.3.2 Ada 95 Reference Manual

9.3.2.1 Ada 2005 Reference Manual

9.3.3 Ada Quality and Style Guide
- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

10 Keywords: array

This keyword is used in the declaration of array types and anonymous array objects.

10.1 See also

10.1.1 Wikibook
11 Keywords: at

This keyword is used in representation clauses.

11.1 See also

11.1.1 Wikibook

11.1.2 Ada Reference Manual

11.1.3 Ada Quality and Style Guide

12 Keywords: begin

The **begin** keyword in Ada is almost synonymous with the { in C/C++/Java. In the paradigm of structured programming languages, a keyword or symbol is often needed to delimit the beginning of the body from the element (i.e. if, elsif, for loop, while loop) to which it pertains.

For the case of subprograms, packages, tasks and blocks this word is **begin**.

For example
The `begin` keyword is always closed with the `end` keyword. The opposite is not true in Ada.

12.1 See also

12.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords
- Ada Programming/Tasking

12.1.2 Ada Reference Manual


12.1.3 Ada Quality and Style Guide


13 Keywords: body

This keyword is used to define the body (implementation) of the following units:

- Packages
- Tasks and protected units

13.1 See also

13.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords


13.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adai.org/docs/95style/html/sec_3/3-1-3.html)

14 Keywords: case

This keyword is used in case statements and variant record definitions.

14.1 See also

14.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords


14.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adai.org/docs/95style/html/sec_3/3-1-3.html)

15 Keywords: constant

This keyword is used to declare constant objects and named numbers.

15.1 See also

15.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

15.1.2 Ada Reference Manual


15.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adai.org/docs/95style/html/sec_3/3-1-3.html)
16 Keywords: declare

This keyword is used to mark beginning of the declarative part of a block.

```
Block_Name :
    declare
     A_Variable : The_Type;
begin
    Use A_Variable
end Block_Name;
```

The `Block_Name` is optional.

16.1 See also

16.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

16.1.2 Ada 95 Reference Manual

- 5.6 Block Statements (http://www.adaic.com/standards/95lrm/html/RM-5-6.html)
- Annex P (informative) Syntax Summary


- 5.6 Block Statements (http://www.adaic.com/standards/05rm/html/RM-5-6.html)
  (Annotated (http://www.adaic.com/standards/05aarm/html/AA-5-6.html))
- Annex P (informative) Syntax Summary
  (Annotated (http://www.adaic.com/standards/05aarm/html/AA-P.html))

16.1.4 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

17 Keywords: delay

17.1 Delay statements

17.1.1 Relative delays
This language feature will be made available in the forthcoming Ada 2005 standard.

The Ravenscar profile forbids the use of the relative delay statement:

```ada
pragma Restrictions (No_Relative_Delay)
```

### 17.1.2 Absolute delays

```ada
Delay until Next_Time;
```

### 17.2 Select statements

#### 17.2.1 Delay alternative

Selective accept:

```ada
select
  when Cond =>
    accept_statement
  statements
or
  delay_alternative
end select;
```

#### 17.2.2 Timed entry call

```ada
select
  entry_call_alternative
or
  delay_alternative
end select;
```

#### 17.2.3 Asynchronous transfer of control

```ada
select
  delay until Abort_Time;
  statements
then abort
  abortable_part
end select;
```

### 17.3 See also

#### 17.3.1 Wikibook

- Ada Programming
- Ada Programming/Keywords
- Ada Programming/Tasking
- Ada Programming/Pragmas/Restrictions/No_Relative_Delay
17.3.2 Ada Reference Manual


17.3.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)
- 6.1.7 Delay Statements (http://www.adaic.org/docs/95style/html/sec_6/6-1-7.html)

18 Keywords: delta

18.1 Description

This keyword is used in the declaration of fixed point types. Delta represents the absolute error bound of a fixed point type; that is, delta gives a hint to the compiler how to choose the smallest incremental value of the type, called the Small of the type. This Small value is accessible via the attribute T'Small.

For Ordinary fixed point types, it is by default a power of two not greater than the delta; it is specifiable by an attribute clause to be any value not greater than the delta.

For decimal fixed point types, the Small value is equal to the delta, which of course must be a power of ten.

18.2 Example

```ada
type Fixed_Point is delta 0.01 digits 10; -- A decimal fixed point type
My_Fixed_Point : Fixed_Point := 0.0;
...
My_Fixed_Point := 0.1;  -- Ok
My_Fixed_Point := 0.02; -- Ok
My_Fixed_Point := 0.001; -- Oops - this value is too small.
                      -- It will be represented by zero.
```

18.3 See also
18.3.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

18.3.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

18.3.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

19 Keywords: digits

This keyword is used in the declaration of floating point types and fixed point types.

19.1 See also

19.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords


- Annex P (informative) Syntax Summary

19.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

20 Keywords: do

This keyword is used in accept statements.

20.1 See also

20.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords
20.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

20.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

21 Keywords: else

This keyword is used in:

- The 'if' statements
- The or else shortcut operator
- Conditional entry calls in tasks.

21.1 See also

21.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords


- Annex P (informative) Syntax Summary

21.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

22 Keywords: elsif

This keyword is used in 'if' statements.

22.1 See also

22.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

22.1.2 Ada Reference Manual
22.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.ada.org/docs/95style/html/sec_3/3-1-3.html)

23 Keywords: end

This keyword is used to mark the end of the following constructs:

- Control statements
- Subprogram bodies
- Records
- Package specifications and bodies
- Tasks, protected objects, accept statements and select statements
- Blocks

23.1 See also

23.1.1 WikiBook

- Ada Programming
- Ada Programming/Keywords

23.1.2 Ada Reference Manual


23.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.ada.org/docs/95style/html/sec_3/3-1-3.html)

24 Keywords: entry

This keyword is used to define a rendezvous in the context of tasking.

24.1 See also

24.1.1 WikiBook

- Ada Programming
24.1.2 Ada Reference Manual


24.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

25 Keywords: exception

This keyword is used in exception declarations and exception handlers.

25.1 See also

25.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

25.1.2 Ada Reference Manual


25.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

26 Keywords: exit

This keyword is used to exit the current or named loop, conditionally or unconditionally.

26.1 See also

26.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

27 Keywords: for

This keyword is used in:

- For loops
- Representation clauses
- Attribute definition clauses

27.1 See also

27.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

27.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

27.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

28 Keywords: function

This keyword is used in:

- the specification and body of functions, a kind of subprograms.
- access to subprogram type declarations.

28.1 See also

28.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

- Annex P (informative) Syntax Summary

28.1.3 Ada Quality and Style Guide


29 Keywords: generic

This keyword is used in generic units declarations.

29.1 See also

29.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

29.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

29.1.3 Ada Quality and Style Guide


30 Keywords: goto

This keyword "goto" is used in the goto statement.

For details see the chapter Ada Programming/Control - goto.

30.1 See also

30.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

30.1.2 Ada Reference Manual

30.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

31 Keywords: if

This keyword is used in 'if' statements.

31.1 See also

31.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords

31.1.2 Ada Reference Manual


31.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

32 Keywords: in

This keyword is used in:

- The in and in out mode of subprograms parameters.
- The membership test.

32.1 See also

32.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords
32.1.2 Ada 95 Reference Manual


32.1.4 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

33 Keywords: interface

Used to declare an interface. Interfaces are the most important innovation in Ada 2005. The basic semantic has been taken from the interface concept of the Java programming language. Interface inheritance is a reduced form of multiple inheritance.

33.1 Interface types

There are various forms of interfaces available:

33.1.1 Normal interface

The normal interface is declare as:

```ada
type Printable is interface;
```

33.1.2 Limited interface

Can be used as interface to limited tagged type.

```ada
type Printable is limited interface;
```

33.1.3 Synchronized interface
All synchronized interfaces are also limited.

### 33.1.4 Task interface

Can be used as an interface to a task type.

```ada
type Printable is synchronized interface;
```

All task interfaces are also synchronized.

### 33.1.5 Protected interface

Can be used as an interface to a protected type.

```ada
type Printable is protected interface;
```

All protected interfaces are also synchronized.

### 33.2 See also

#### 33.2.1 Wikibook

- Ada Programming
- Ada Programming/Keywords
- Ada Programming/Object Orientation
- Ada Programming/OO

#### 33.2.2 Ada Reference Manual

- 2.9 Reserved Words  
  (Annotated)
- Annex P Syntax Summary
  (Annotated)

#### 33.2.2.1 Ada 2005

- 3.9.4 Interface Types  
  (Annotated)

#### 33.2.3 Ada Quality and Style Guide

- 3.1.3 Capitalization

### 34 Keywords: is

The keyword `is` separates the name of an entity from its definition.
34.1 In type declarations

```ada
type Day_Of_Month is new range 1 .. 31;
```

See Ada Programming/Types and Ada Programming/Subtypes.

34.2 In subtype declarations

```ada
subtype Day_Of_February is Day_Of_Month range 1 .. 29;
```

See Ada Programming/Subtypes.

34.3 In package declarations

```ada
package My_Package is
    ... -- declarations
end My_Package;
```

See Ada Programming/Packages.

34.4 In procedure and function declaration

```ada
procedure My_Procedure is
    ... -- declarations
begin
    ... -- sentences
end My_Procedure;
```

See Ada Programming/Subprograms.

34.5 In generic instantiations

```ada
package Day_Of_Month_IO is
    new Ada.Text_IO.Integer_IO (Num => Day_Of_Month);
```

See Ada Programming/Generics.

34.6 See also

34.6.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

34.6.2 Ada Reference Manual

34.6.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

35 Keywords: limited

This keyword is used to define limited types.

35.1 See also

35.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

35.1.2 Ada Reference Manual


35.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

36 Keywords: loop

This keyword is used in loop statements.

36.1 See also

36.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

36.1.2 Ada Reference Manual


36.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)
37 Keywords: mod

This keyword is used in the mod operator and in the declaration of modular types.

37.1 See also

37.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords

37.1.2 Ada 95 Reference Manual
- Annex P (informative) Syntax Summary

- Annex P (informative) Syntax Summary
  (Annotated (http://www.adaic.com/standards/05aarm/html/AA-P.html))

37.1.4 Ada Quality and Style Guide
- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

38 Keywords: new

38.1 Declaration

38.1.1 In type derivation

```plaintext
type Day_Of_Month is new Integer range 1 .. 31;
```

See Declaring new Types.

38.1.2 In type extension

```plaintext
type Derived_Tagged_Type is new Parent_Tagged_Type with
record
  New_Data : Data;
end record;
```

See Ada Programming/Object Orientation.
38.1.3 In instantiation of generics

```ada
package Day_Of_Month_IO is
  new Ada.Text_IO.Integer_IO (Day_Of_Month);
end Day_Of_Month_IO;
```

See Generics.

38.2 Allocator

```ada
type Day_Of_Month_Access is access Day_Of_Month;
X : Day_Of_Month_Access := new Day_Of_Month'(5);
```

See Creating object in a storage pool.

38.3 See also

38.3.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

38.3.2 Ada Reference Manual


38.3.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

39 Keywords: not

This keyword is used in:

- Logical negation operator
- Negative membership test: not in

39.1 See also

39.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords
39.1.2 Ada 95 Reference Manual

- 4.5.6 Highest Precedence Operators (http://www.adaic.com/standards/95lrm/html/RM-4-5-6.html)


- 4.4 Expressions (http://www.adaic.com/standards/05rm/html/RM-4-4.html)
- 4.5.6 Highest Precedence Operators (http://www.adaic.com/standards/05rm/html/RM-4-5-6.html)

39.1.4 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

40 Keywords: null

This keyword is used for:

- Null access value
- Null record definition
- Null variant part in a record definition
- Null statement

40.1 See also

40.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

40.1.2 Ada Reference Manual


40.1.3 Ada Quality and Style Guide
3.1.3 Capitalization

41 Keywords: of

This keyword is used in array types declarations.

41.1 See also

41.1.1 Wikibook

Ada Programming
Ada Programming/Keywords

41.1.2 Ada Reference Manual

2.9 Reserved Words
(Annotated)
Annex P Syntax Summary
(Annotated)

41.1.3 Ada Quality and Style Guide

3.1.3 Capitalization

42 Keywords: or

42.1 Logical operator

42.1.1 Boolean operator

\[
X : \text{Boolean} := A < 10 \textbf{ or } A > 20;
\]

42.1.2 Boolean shortcut operator

In the below example the function \(G\) is only called when \(F(X)\) returns the value \(\text{False}\).

\[
\text{if } F(X) \textbf{ or else } G(Y) \text{ then}
  \begin{align*}
  & \text{Walk_The_Dog;}
  \end{align*}
\text{end if;}
\]

This shortcut operator is sometimes used to speed up the evaluation of boolean expressions, but the Ada Style Guide recommends to compare the performance of both forms before switching one to the other. In general, it is good idea to use \textbf{or else} in sake of performance only when the second expression involves a function call.

The \textbf{or else} form is also used when the second expression is known to raise an exception.
unless the first expression is \textit{False}.

Unlike C/C++, Ada short-cut operators are not the standard way to evaluate boolean expressions. This is because Ada is designed to do by default what is generally safer, but lets the programmer request a different behaviour.

42.1.3 Boolean operator on arrays

The or operator is applied to each pair of boolean elements from the left and right arrays. The result has the same bounds than the left operand.

42.1.4 Bitwise operator

The operator \texttt{or} could be used with modular types to perform bitwise operations.

42.2 Select statement

42.2.1 alternative

\textit{See Ada Programming/Tasking\#Selective\_waiting.}

42.2.2 delay

\textit{See Ada Programming/Tasking\#Timeout.}

42.3 See also

42.3.1 Wikibook

\begin{itemize}
  \item Ada Programming
  \item Ada Programming/Keywords
  \item Ada Programming/Operators
\end{itemize}

42.3.2 Ada 95 Reference Manual

\begin{itemize}
  \item 4.5.1 Logical Operators and Short-circuit Control Forms (http://www.adaic.com/standards/95lrm/html/RM-4-5-1.html) (Annotated (http://www.adaic.com/standards/95aarm/html/AA-4-5-1.html))
\end{itemize}

42.3.3 Ada 2005 Reference Manual

\begin{itemize}
  \item 2.9 Reserved Words (http://www.adaic.com/standards/95lrm/html/RM-2-9.html)
\end{itemize}
42.3.4 Ada Quality and Style Guide

- 3.1.3 Capitalization
- 5.5.5 Short Circuit Forms of the Logical Operators
- 10.5.2 Short-Circuit Operators
- 10.6.3 Bit Operations on Modular Types

43 Keywords: others

43.1 Summary

The keyword **others** is used to define the remaining options in a list of options:

- In exception handlers
- In case statements
- In variant records
- In array initialisations

43.2 See also

43.2.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

43.2.2 Ada Reference Manual

- 2.9 Reserved Words
- Annex P Syntax Summary

43.2.3 Ada Quality and Style Guide

- 3.1.3 Capitalization

44 Keywords: out
This keyword is used in the **out** and **in out** mode of subprograms parameters.

### 44.1 See also

#### 44.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords

#### 44.1.2 Ada 95 Reference Manual

#### 44.1.3 Ada 2005 Reference Manual

#### 44.1.4 Ada Quality and Style Guide
- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

### 45 Keywords: overriding

The keyword **overriding** can be used to explicitly declare a procedure or function as an overriding subprogram. There are two forms available:

```ada
overriding
procedure Overriding_Procedure (This: in Class_Type);

not overriding
procedure Not_Overriding_Procedure (This: in Class_Type);
```

The first declares the procedure as overriding the second does not.

### 45.1 See also

#### 45.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords
45.1.2 Ada Reference Manual


45.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

46 Keywords: package

This keyword is used in regular packages (specification and body), in generic packages (specification and body), and in generic formal package parameters.

46.1 See also

46.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

46.1.2 Ada 95 Reference Manual


46.1.4 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

47 Keywords: pragma

This keyword is used in pragma statements.

47.1 See also

47.1.1 Wikibook
47.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary  

47.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

48 Keywords: private

This keyword is used to declare:

- Private types
- The private part in a package specification
- Private child packages

48.1 See also

48.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

48.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary  

48.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

49 Keywords: procedure

This keyword is used in:

- the specification and body of procedures, a kind of subprograms.
- access to subprogram type declarations.
- Procedures as parameters in generic units.
49.1 See also

49.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

49.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

49.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

50 Keywords: protected

This keyword is used in the definition of protected types.

50.1 See also

50.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

50.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

50.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

51 Keywords: raise

This keyword is used for raising a named exception or reraising the current exception in an exception handler.

51.1 See also

51.1.1 Wikibook
52 Keywords: range

52.1 Description

The keyword `range` is used in the following type definitions.

- Defining the first and last element in an integer range.
- Defining the first and last element in a Modulus subtype.
- Defining the first and last element in an enumeration subtype.
- Defining the minimum range of a floating point type.
- Defining the minimum range of a fixed point type.
- Defining the range of an array type.

It is also used within the following statements:

- Specify the range of a for loop.

52.2 See also

52.2.1 Wikibook

- Ada Programming
- Ada Programming/Keywords
- Ada Programming/Types/range
- Ada Programming/Types/Enumeration
- Ada Programming/Types/digits
- Ada Programming/Types/delta
- Ada Programming/Types/array
- Ada Programming/Control

52.2.2 Ada Reference Manual


3.6 Array Types

- 3.6.1 Index Constraints and Discrete Ranges

4.4 Expressions

52.2.3 Ada Quality and Style Guide

3.1.3 Capitalization

53 Keywords: record

This keyword is used in record types declarations.

53.1 See also

53.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

53.1.2 Ada Reference Manual

- 2.9 Reserved Words
- Annex P Syntax Summary

53.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization

54 Keywords: rem

54.1 Operator rem

The `rem` keyword is used as the remainder operator, that is, the remainder of the signed integer division. The following formula applies:

\[ A = (A / B) \times B + (A \text{ rem } B) \]

54.2 See also

54.2.1 Wikibook

- Ada Programming
54.2.2 Ada 95 Reference Manual

- Annex P (informative) Syntax Summary
- 4.5.5 Multiplying Operators
  (http://www.adaic.com/standards/95lm/html/RM-4-5-5.html)
  (Annotated (http://www.adaic.com/standards/95aarm/html/AA-4-5-5-5.html))

54.2.3 Ada 2005 Reference Manual

- Annex P (informative) Syntax Summary
  (Annotated (http://www.adaic.com/standards/05aarm/html/AA-P.html))
- 4.5.5 Multiplying Operators (http://www.adaic.com/standards/05rm/html/RM-4-5-5.html)
  (Annotated (http://www.adaic.com/standards/05aarm/html/AA-4-5-5.html))

54.2.4 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

55 Keywords: renames

This keyword is used in renaming declarations.

55.1 See also

55.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

55.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

55.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

56 Keywords: requeue
This keyword is used for requeue statements.

56.1 See also

56.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

56.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

56.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

57 Keywords: return

This keyword is used in:

- Functions specifications for declaring the returned type.
- In function bodies for returning a value to the caller.
- In procedure bodies for returning the control to the caller.

57.1 See also

57.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

57.1.2 Ada 95 Reference Manual

- 6.5 Return Statements (http://www.adaic.com/standards/95lrm/html/RM-6-5.html)
  (Annotated (http://www.adaic.com/standards/95aarm/html/AA-6-5.html))
- Annex P (informative) Syntax Summary


- 6.5 Return Statements (http://www.adaic.com/standards/05rm/html/RM-6-5.html)
  (Annotated (http://www.adaic.com/standards/05aarm/html/AA-6-5.html))
- Annex P (informative) Syntax Summary
  (Annotated (http://www.adaic.com/standards/05aarm/html/AA-P.html))
57.1.4 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

58 Keywords: reverse

This keyword is used in for loops to specify that the loop is performed in reverse order:

```ada
for I in reverse 1 .. 10 loop
    Ada.Integer_Text_IO.Put (I);
    Ada.Text_IO.New_Line;
end loop;
```

Outputs:

```
10 9 8 7 6 5 4 3 2 1
```

58.1 See also

58.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

58.1.2 Ada Reference Manual

- Annex P Syntax (informative) Summary

58.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

59 Keywords: select

This keyword is used in selective waiting in tasks.
59.1 See also

59.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords

59.1.2 Ada Reference Manual

59.1.3 Ada Quality and Style Guide
- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

60 Keywords: separate

This keyword is used to declare subunits.

60.1 See also

60.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords

60.1.2 Ada Reference Manual

60.1.3 Ada Quality and Style Guide
- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

61 Keywords: subtype

61.1 Summary

The keyword **subtype** is used to declare and name a new subtype. For more information on types and subtypes read Ada Programming/Subtypes.

61.2 See also
61.2.1 Wikibook

- Ada Programming
- Ada Programming/Keywords
- Ada Programming/Types
- Ada Programming/Subtypes

61.2.2 Ada Reference Manual


61.2.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

62 Keywords: synchronized

This language feature will be made available in the forthcoming Ada 2005 standard.

The **synchronized** keyword is used to declare that the interface has to be implemented by task types or protected types. The syntax is:

```pascal
type Printable is synchronized interface;
```

62.1 See also

62.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

62.1.2 Ada Reference Manual


62.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

63 Keywords: tagged

This keyword is used in object-oriented type declarations (tagged types).
63.1 See also

63.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords

63.1.2 Ada Reference Manual
- Annex P (informative) Syntax Summary

63.1.3 Ada Quality and Style Guide
- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

64 Keywords: task

The keyword task is used to define either a task or task type. See Ada Programming/Tasking for more information on the Ada tasking system.

64.1 See also

64.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords
- Ada Programming/Tasking

64.1.2 Ada Reference Manual
- Annex P (informative) Syntax Summary

64.1.2.1 Ada 95
- Section 9: Tasks and Synchronization

64.1.2.2 Ada 2005
  (Annotated (http://www.adaic.com/standards/05aarm/html/AA-3-9-4.html))
- Section 9: Tasks and Synchronization
64.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

65 Keywords: terminate

This keyword is used in task termination.

65.1 See also

65.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

65.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

65.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

66 Keywords: then

This keyword is used in:

- 'if' statements
- 'and then' shortcut operator
- Asynchronous select

66.1 See also

66.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

66.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

66.1.3 Ada Quality and Style Guide
67 Keywords: type

The keyword **type** is used to declare a new type and name the first subtype. For more information on types and subtypes read Ada Programming/Subtypes.

### 67.1 See also

#### 67.1.1 Wikibook
- Ada Programming
- Ada Programming/Keywords
- Ada Programming/Types
- Ada Programming/Subtypes

#### 67.1.2 Ada Reference Manual
- 2.9 Reserved Words (Annotated)
- Annex P Syntax Summary (Annotated)
- 3.2 Types and Subtypes (Annotated)

#### 67.1.3 Ada Quality and Style Guide
- 3.1.3 Capitalization

68 Keywords: until

The keyword **until** is used only in delay until statements.

### 68.2 See also

#### 68.2.1 Wikibook
- Ada Programming
- Ada Programming/Keywords

#### 68.2.2 Ada Reference Manual
- 2.9 Reserved Words (Annotated)
- Annex P Syntax Summary (Annotated)
69 Keywords: use

69.1 Use clause

69.1.1 At library level

This use clause make the content of a package immediately visible.

```ada
with Ada.Text_IO; use Ada.Text_IO;

procedure Hello is
begin
  Put_Line("Hello, world!");
  New_Line;
  Put_Line("I am an Ada program with package use.");
end Hello;
```

If readability is your main concern then you should avoid this type of use clause. However it is needed to resolve a name clash between the packages which are imported and the package currently compiled.

69.1.2 At declaration level

```ada
with Ada.Text_IO;

procedure Hello is
  use Ada.Text_IO;
begin
  Put_Line("Hello, world!");
  New_Line;
  Put_Line("I am an Ada program with package use.");
end Hello;
```

69.1.3 For types
with Ada.Text_IO;

procedure Hello is
  use type Ada.Text_IO.File_Type
begin
  Ada.Text_IO.Put_Line("Hello, world!");
  Ada.Text_IO.New_Line;
  Ada.Text_IO.Put_Line("I am an Ada program with package use.");
end Hello;

69.2 Representation clause

69.2.1 For attributes

```ada
type Day_Of_Month is range 1 .. 31;
for Day_Of_Month'Size use 8;
```

69.2.2 For records

69.2.3 For enumerations

69.3 See also

69.3.1 Wikibook
- Ada Programming
- Ada Programming/Keywords

69.3.2 Ada Reference Manual

69.3.3 Ada Quality and Style Guide
- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)
70 Keywords: when

This keyword is used in:

- case statements,
- variant record definitions,
- exception handlers
- loops exit conditions.

70.1 See also

70.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

70.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

70.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

71 Keywords: while

This keyword is used in the while loops.

71.1 See also

71.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

71.1.2 Ada Reference Manual

- Annex P (informative) Syntax Summary

71.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)
72 Keywords: with

This keyword is used in:

- With clauses
- Extension declarations in OOP
- Subprogram and package parameters in generics

72.1 See also

72.1.1 Wikibook

- Ada Programming
- Ada Programming/Keywords

72.1.2 Ada Reference Manual


72.1.3 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adaic.org/docs/95style/html/sec_3/3-1-3.html)

73 Keywords: xor

73.1 Logical operator

73.1.1 Boolean operator

\[
X : \text{Boolean} := A = 10 \text{ xor } B = 10;
\]

73.1.2 Boolean operator on arrays

The xor operation is applied to each boolean inside the array.

\[
\begin{align*}
\text{type} & \text{ Day_Of_Month is range } 1 \ldots 31; \\
\text{type} & \text{ Month_Array is array (Day_Of_Month) of Boolean;} \\
X & \text{ : Month_Array := Function_1;} \\
Y & \text{ : Month_Array := Function_2;} \\
Z & \text{ : Month_Array := X xor Y;} \\
\end{align*}
\]

73.1.3 Bitwise operator

The operator xor could be used with modular types and also with boolean arrays to perform bitwise operations.
73.2 See also

73.2.1 Wikibook

- Ada Programming
- Ada Programming/Keywords
- Ada Programming/Operators

73.2.2 Ada 95 Reference Manual


73.2.3 Ada 2005 Reference Manual


73.2.4 Ada Quality and Style Guide

- 3.1.3 Capitalization (http://www.adai.org/docs/95style/html/sec_3/3-1-3.html)
- 10.6.3 Bit Operations on Modular Types (http://www.adai.org/docs/95style/html/sec_10/10-6-3.html)

Von "http://en.wikibooks.org/wiki/Ada_Programming/All_Keywords"


- All text is available under the terms of the GNU Free Documentation License (see Copyrights for details).
- Wikibooks® is a registered trademark of the Wikimedia Foundation, Inc.
- Datenschutz
- Über Wikibooks
- Impressum