Ada REFERENCE CARD

bold	Ada keyword	*italic*	Ada 95
[ ]	Optional term	{ }	Repeatable
|	Alternative	...	Identical term

ATTRIBUTES

S - subtype E - entry declaration or exception
T - task X - object
P - program A - discriminated type or array
R - record D - library-level declaration

P'Access Access to subprogram

X'Access Access to object

X'Address Address of the first of the storage elements allocated to object, program unit, or label

S'*Adjacent* Adjacent machine number of argument towards the second floating point argument.

S'Aft The number of decimal digits needed after the decimal point to accommodate the delta

X'*Alignment* Alignment of object

S'Base Denotes the base unconstrained subtype

S'*Bit_Order* Record subtype bit ordering (type System.Bit_Order)

P'*Body_Version* Version of the compilation unit that contains the body

T'Callable True when the task denoted by T is callable

E'*Caller* Value of the type Task_ID that identifies the task whose call is now being serviced

S'*Ceiling* Smallest (most negative) integral value greater than or equal to argument

S'*Class* Subtype of the class-wide type

X'*Component_Size* Size in bits of components of the array subtype or object

S'*Compose* Combine fraction and integer arguments into a floating point subtype

A'Constrained True if discriminated type denotes a constant, a value, or a constrained variable

S'*Copy_Sign* Result whose magnitude is that of float Value and whose sign is that of Sign

E'Count Number of calls presently queued on the entry

S'*Definite* True if the actual subtype of a a formal indefinite subtype is definite

S'Delta The delta (universal_real) of the fixed point subtype

S'*Denorm* True if every value expressible in canonical form with an an exponent of T'Machine_Emin

S'Digits Number of digits of the decimal fixed point subtype

S'Digits Number of decimal mantissa digits for floating point subtype

S'*Exponent* Normalized exponent of the floating point argument

S'*External_Tag* An external string representation of the tagged type

A'First(N) Lower bound of N-th index of [constrained] array type

A'First Lower bound of first index of [constrained] array type

S'First Lower bound of the range of scalar subtype

R.C'First_Bit Bit offset, from the start of the first of the storage elements occupied by C, of the first bit occupied by C

S'*Floor* Largest integral value less than or equal to the argument

S'Fore Minimum number of characters needed before the decimal point

S'*Fraction* Decompose floating point argument into fractional part

E'*Identity* Unique identity of the exception

T'*Identity* Value of type Task_ID identifying the task

S'Image Image of the value of argument as a String

S'*Input* Reads and returns one value from the Stream argument

A'Last(N) Upper bound of N-th index range of [constrained] array type

A'Last Upper bound of first index range of [constrained] array type

S'Last Upper bound of the range of scalar subtype

R.C'Last_Bit Bit offset, from the start of the first of the storage elements occupied by C, of the last bit occupied by C

S'*Leading_Part* The leading part of floating point value with number of radix digits given by second argument

A'Length(N) Number of values of the N-th index range of [constrained] array type

A'Length Number of values of the first index range of [constrained] array type

S'*Machine* Machine representation of floating point argument

S'Machine_Emax Largest (most positive) value of floating point exponent

S'Machine_Emin Smallest (most negative) value of floating point exponent

S'Machine_Mantissa Number of digits in machine representation of mantissa

S'Machine_Overflows True if numeric overflow detected for fixed or floating point

S'Machine_Radix Radix of machine representation of the fixed or floating point

S'Machine_Rounds True if rounding is performed on inexact results of the fixed or floating point

S'*Max* The greater of the values of the two scalar arguments

S'*Max_Size_In_Storage_Elements* Maximum value for Size_In_Storage_Elements that will be requested via Allocate

S'*Min* The lesser of the values of the two scalar arguments

S'*Model* Model number of floating point type

S'*Model_Emin* Model number version of S''Machine_Emin

S'*Model_Epsilon* Absolute difference between the model number 1.0 and the next model number above for subtype.

S'*Model_Mantissa* Model number version of S''Machine_Mantissa

S'*Model_Small* Smallest positive model number of subtype

S'*Modulus* The modulus (universal_integer) of the modular subtype

S'*Output* Writes the value of Item to Stream, including any bounds or discriminants

D'*Partition_ID* Identifies the partition in which D was elaborated

S'Pos Position of the value of the discrete subtype argument

R.C'Position Same as R.C'Address - R'Address for component C

S'Pred Predecessor of the argument

A'Range(N) Equivalent to the range A'First(N) .. A'Last(N)

A'Range Equivalent to the range A'First .. A'Last

S'Range Equivalent to the range S'First .. S'Last

S'*Read* Reads the value of Item from the Stream argument

S'*Remainder* Remainder after dividing the first floating point argument by its second.

S'*Round* Fixed-point value obtained by rounding X (away from 0, if X is midway between two values)

S'*Rounding* Floating-point integral value nearest to X, rounding away from zero if X lies exactly halfway between two integers

S'Safe_First The lower bound of the safe range

S'Safe_Last The upper bound of the safe range

S'*Scale* Position of the fixed-point relative to the rightmost significant digits of values of subtype

S'*Scaling* Scaling by a power of the hardware radix.

S'*Signed_Zeros* True if positive and negative signed zeros are representable

S'Size Size in bits of objects instantiated from subtype

X'Size Size in bits of the representation of the object

S'Small Small of the fixed-point type

S'*Storage_Pool* Storage pool of the access subtype

S'*Storage_Size* Number of storage elements reserved for the storage pool

T'Storage_Size	Number of storage elements reserved for the task
S'Succ	Successor of the argument
S[X]'*Tag*	The tag (type Tag) of the [class-wide] tagged type
T'Terminated	True if the task denoted by T is terminated
S'*Truncation*	The value Ceiling(X) when X is negative, else Floor(X)
S'*Unbiased_Rounding*	Integral value nearest to X, rounding toward the even integer if X lies exactly halfway between two integers.
X'*Unchecked_Access*	Same as X'Access but lacks accessibility rules/checks
S'Val	Value of the discrete subtype whose position number equals the value of argument
X'*Valid*	True if and only if the scalar object denoted by X is normal and has a valid representation
S'Value	Returns a value of the subtype given an image of the value as a String argument
P'*Version*	The version of the compilation unit that contains the declaration
S'*Wide_Image*	Image of the value of argument as a Wide_String
S'*Wide_Value*	Returns a value given an image of the value as a Wide_String argument
S'*Wide_Width*	Maximum length of Wide_String returned by S'Image
S'Width	Maximum length of String returned by S'Image
S'*Write*	Writes the value of Item to Stream argument

PRAGMAS

pragma *All_Calls_Remote*[(library_unit_name)];
pragma *Asynchronous*(local_name);
pragma *Atomic*(local_name);
pragma *Atomic_Components*(array_local_name);
pragma *Attach_Handler*(handler_name, expression);
pragma Controlled(first_subtype_local_name);
pragma *Convention*([Convention =>] convention_identifier,
[Entity =>] local_name);
pragma *Discard_Names*[([On =>] local_name)];
pragma Elaborate(library_unit_name{, ...});
pragma *Elaborate_All*(library_unit_name{, ...});
pragma *Elaborate_Body*[(library_unit_name)];
pragma *Export*([Convention =>] convention_identifier,
[Entity =>] local_name [,
[External_Name =>] string_expression] [,
[Link_Name =>] string_expression]);
pragma *Import*([Convention =>] convention_identifier,
[Entity =>] local_name [,
[External_Name =>] string_expression] [,
[Link_Name =>] string_expression]);
pragma Inline(name {, ...});
pragma *Inspection_Point*[(object_name {, ...})];
pragma *Interrupt_Handler*(handler_name);
pragma *Interrupt_Priority*[(expression)];
pragma *Linker_Options*(string_expression);
pragma List(identifier);
pragma *Locking_Policy*(policy_identifier);
pragma *Normalize_Scalars*;
pragma Optimize(identifier);
pragma Pack(first_subtype_local_name);
pragma Page;
pragma *Preelaborate*[(library_unit_name)];
pragma Priority(expression);
pragma *Pure*[(library_unit_name)];
pragma *Queuing_Policy*(policy_identifier);
pragma *Remote_Call_Interface*[(library_unit_name)];
pragma *Remote_Types*[(library_unit_name)];
pragma *Restrictions*(restriction{, ...});
pragma *Reviewable*;
pragma *Shared_Passive*[(library_unit_name)];
pragma *Storage_Size*(expression);
pragma Suppress(identifier [, [On =>] name]);
pragma *Task_Dispatching_Policy*(policy_identifier);
pragma *Volatile*(local_name);
pragma *Volatile_Components*(array_local_name);

STANDARD LIBRARY

package Standard

Boolean	True or False
Integer	Implementation defined
Natural	Integers >= 0
Positive	Integers > 0
Float	Implementation defined
Character	*8*-bit ASCII
Wide_Character	16-bit ISO 10646
String	Array of Characters
Duration	Time
Constraint_Error	Predefined Exception
Program_Error	Predefined Exception
Storage_Error	Predefined Exception
Tasking_Error	Predefined Exception

package *Ada*

- *Asynchronous_Task_Control*
- *Calendar*
- *Characters*
  - *Handling*
  - *Latin_1*
- *Command_Line*
- *Decimal*
- *Direct_IO*
- *Dynamic_Priorities*
- *Exceptions*
- *Finalization*
- *Float_Text_IO*
- *Integer_Text_IO*
- *Interrupts*
  - *Names*
  - *IO_Exceptions*
- *Numerics*
  - *Complex_Elementary_Functions*
  - *Complex_Types*
  - *Discrete_Random*
  - *Elementary_Functions*
  - *Float_Random*
  - *Generic_Complex_Elementary_Functions*
  - *Generic_Complex_Types*
  - *Generic_Elementary_Functions*
- *Real_Time*
- *Sequential_IO*
- *Storage_IO*
- *Streams*
  - *Stream_IO*
- *Strings*
  - *Bounded*
  - *Fixed*
  - *Maps*
    - *Constants*
  - *Unbounded*
  - *Wide_Bounded*
  - *Wide_Fixed*
  - *Wide_Maps*
    - *Wide_Constants*
  - *Wide_Unbounded*
- *Synchronous_Task_Control*
- *Tags*
- *Task_Attributes*
- *Task_Identification*
- *Text_IO*
  - *Complex_IO*
  - *Text_Streams, etc*
- *Unchecked_Conversion*
- *Unchecked_Deallocation*
- *Wide_Text_IO*
  - *Complex_IO*
  - *Text_Streams, etc*

package *Interfaces*

- *C*
  - *Pointers*
  - *Strings*
- *COBOL*
- *Fortran*

package *System*

- *Address_To_Access_Conversions*
- *Machine_Code*
- *RPC*
- *Storage_Elements*
- *Storage_Pools*