Ada 2012

ADA LOG

Ada steaming ahead: New 2012 features

J-P. Rosen
Adalog
www.adalog.fr

Syntactic sugar (tastes good!)

- For .. of

```
for I in Tab'Range loop
   Tab (I) := Tab (I) +1;
end loop;
```

$\Rightarrow$

```
for Elem of Tab loop
   Elem := Elem +1;
end loop;
```

- Membership with list of values

```
if I in 1..10 | 12 | 20 then ...
```

- if expressions, case expressions

```
X := (if I < 0 then A else B);
```

```
X := (case Color is
         when Red   => 1,
         when Green => 2,
         when Blue  => 4);
```

- predicates

```
X_Is_Prime := (for all  Div in 2..Sqrt (X) => X rem Div /= 0);
X_In_Tab   := (for some Inx in Tab'Range   => Tab (Inx) = X);
```

Syntactic sugar (tastes good!)

- Floating label

```
for I in Tab'Range loop
   ,,,
   if Nothing_Else_To_Do then
      goto Continue;
   end if;
   …
   <<Continue>>
end loop;
```

null;

- Expression functions

```
function Norm (X, Y : Float) return Float is
        (Sqrt (X**2 + Y**2);
```

▶ Allowed in package specifications

Other small sweet goodies

- **use all** *type*
  - Makes all primitive operations (including enumeration litterals) directly visible
  - Default values for discriminants of a limited tagged type
- Default initial value for any (sub)type

```
type Counter is range 0 .. 100
   with Default_Value => 0;
```

- Constant return object
- All records compose for equality
  - Previously : only tagged types

Semantic sour medicine

- Many fixes to obscure corners
  - Accessibility rules
  - Freezing rules
  - …
- Casual users don't have to care
  - If your program doesn't compile any more, it had a bug

Subprograms

- **out** and **in out** modes for functions
  - Functions (not procedures) had only **in** (read-only) parameters
  - Winner by exhaustion
- Protection against aliasing
  - For functions *and procedures* : different **out** or **in out** formal parameters (of an elementary type) are not allowed to refer to the same actual parameter
  - Also in other cases where order of evaluation matters

```
procedure P (X, Y : in out Integer);
function  F (Var  : in out Integer) return Integer;
…
P (V, V);                     -- Illegal !
Pair_Of_Ints := (V, F(V)); -- Illegal !
```

Aspects

- Before:
  - Pragmas, representation clauses, special constructs

```
V : Integer_8;
pragma Atomic (V);
for V'Address use To_Address (16#ADA#);
```

- Now:
  - Unified way of specifying additional properties of any entity

```
    V : Integer_8
      with Atomic,
           Address => To_Address (16#ADA#);
```

More clearly bound to entities, avoids some ambiguities

User defined container features

- Indexing, referencing, iterator
  - It's a bit awkward...
  - Specified by a combination of interfaces and aspects
- All containers have them
  - Can be treated like arrays : indexing (by any type), for.. in.. loops, for.. of.. loops
  - Makes containers *a lot easier* to use
- Not limited to standard containers!

Predefined library

- Internationalization
  - Access to country codes and language codes
- Files and directories
  - relative path, case sensitivity...
- UTF encoding
  - Management of BOMs
  - String conversions
- More containers
  - Bounded forms, indefinite holder
  - Trees and queues
  - Synchronized containers

Tasking

- Multi-cores
  - Package System.Multiprocessors
  - Assignment of task to CPU
  - Dispatching domains (static and dynamic attachment)
- Synchronous barrier
- Time spent in interrupts
- Yield, Yield_to_higher

Programming with contracts

- What is it ?
  - With software components, there is a provider of the component who is different from the user of the component
  - For each provided service, define rights and obligations of the user and of the provider of the service
    - A precondition expresses what is required from the user.
    - A postcondition expresses what is promised by the provider.
    - An invariant is a property that always holds (from the POV of the user).
- These conditions are part of the specification
  - Visible !

Assertions (Ada 2005)

- pragma Assert

```
pragma Assert (Condition, Message);
```

- pragma Assertion_Policy
  - Check : if the condition is false, raise Assertion_Error with the given message
  - Ignore : condition not checked

- Enforce invariants, easily removed for production use

Subtype predicates

- Generalization of the notion of constraint
  - *Static* predicates
    - must be static (!)
    - enjoy many checks at compile time (including full coverage of **case** statements)
  - *Dynamic* predicates
    - no restriction
  - Checked only when Assertion_Policy is Check

```
subtype Even is Integer
   with Dynamic_Predicate => Even mod 2 = 0;

subtype Winter is Month
   with Static_Predicate => Winter in Dec | Jan | Feb;
```

Pre and Postconditions

- On subprograms
  - Pre and Post apply to a single type
  - Pre'Class and Post'Class apply also to descendants
  - Checked only when Assertion_Policy is Check
- Special attributes for post-conditions
  - V'Old : value of V on subprogram entrance
  - F'Result : value returned by function F

```
procedure Update_Person (P : in out Person)
   with Post => P.Sex = P.Sex'Old
                and P.Birth_Date = P.Birth_Date'Old;

function Inc(X: Integer) return Integer
   with Pre  => X /= Integer'Last,
        Post => Inc'Result = X'Old+1;
```

Type invariants

- Only for private types
- Apply only outside the package
  - may be temporarily violated by services inside the package

```
package Places is
   type Disc_Point is private
      with Type_Invariant => Check_In(Disc_Pt);

   function Check_In(D: Disc_Point) return Boolean;
   ...   -- various operations on disc points

private
   type Disc_Point is
      record
         X, Y: Float range -1.0 .. +1.0;
      end record;

   function Check_In (D: Disc_Point) return Boolean is
           (D.X**2 + D.Y**2 <= 1.0)
      with Inline;
end Places;
```

Conclusion

- Fixes and small improvements
- Friendlier for users
- Important additions:
  - Aspects
  - Support for multi-cores
  - Programming by contract
- Additions & improvement to the standard library

Not an earth-shake, continuing improvements

More info: http://www.ada2012.org/