Ada Magica

Christoph Grein

c/o ESG, 81677 München; eMail: Christ-Usch.Grein@T-Online.de

Ada im Paradies - Lösung

Das Problem illustriert einige der komplizierteren Regeln der Parameterübergabe im RM 6.4.1. Wir haben beschränkte und unbeschränkte Ausprägungen eines generischen Pakets:

```
generic
  type Element is private;
package Paket is
  procedure Operation (Parameter: out Element);
end Paket;

type Element (Diskriminante: Diskreter_Typ := Vorbelegung) is …
subtype Eingeschränkt is Element (Einschränkung);

package Unbeschränkte_Ausprägung is new Paket (Element);
package Beschränkte_Ausprägung   is new Paket (Eingeschränkt);
```

Bei der unbeschränkten Ausprägungen müsssen wir natürlich beim Zugriff auch ein uneingeschränktes Objekt verwenden, da wir gewöhnlich die Einschränkung des gelieferten Objekts nicht vorhersagen können. Wenn wir dasselbe jedoch für die beschränkte Ausprägung versuchen, bekommen wir eins auf die Finger geklopft:

```
Objekt: Element;  -- uneingeschränkt
Unbeschränkte_Ausprägung.Operation (Objekt);  -- in Ordnung
Beschränkte_Ausprägung  .Operation (Objekt);  -- Constraint_Error
```

Ada 83 und Ada 95 beschreiben das Verhalten zwar auf unterschiedliche Weise (RM_83 6.4.1(9) bzw. RM_95 6.4.1(16)), doch das Ergebnis bleibt dasselbe: Im ersteren Fall ändert sich die Diskriminante bei der Ausgabe in passender Weise, im letzteren Fall muss sie schon vor dem Aufruf den richtigen Wert haben.

Der Grund für diese Regel? Ich weiß ihn auch nicht – vermutlich kann mit ihr der erzeugte Maschinenkode effizienter gemacht werden durch Vermeidung von Laufzeitprüfungen, was ja ein erklärtes Ziel des Sprachstandards ist. Man beachte auch AARM_95 6.4.1(17e): Die Regeln der Parameterübergabe stellen sicher, dass implizite Initialisierungen nicht verloren gehen.

Die Lösung unseres paradiesischen Problems besteht somit in der Deklaration lokaler Objekte mit der korrekten Diskriminante:

```
procedure Lies_Sünde (Wessen: in     Mensch;
                      Welche: in     Nummer;
                      Sünde :    out Eintrag) is
begin
  case Wessen is
    when Adam => declare
                   Lokal: Adams_Eintrag;
                 begin
                   Adams_Sündenregister.Lies (Welche, Lokal);
                   Sünde := Lokal;
                 end;
    when Eva  => declare
                   Lokal: Evas_Eintrag;
                 begin
                   Evas_Sündenregister.Lies (Welche, Lokal);
                   Sünde := Lokal;
                 end;
  end case;
end Lies_Sünde;
```

Hierzu passt ganz gut das folgende Problem mit der Parameterübergabe bei generischen Ausprägungen (RM_95 12.5), bei dem tatsächlich verschiedene Compiler (noch) unterschiedlicher Meinung sind.

Gegeben seien folgende Deklarationen:

```
generic
  type Gezeigt is private;
  type Zeiger  is access Gezeigt;
package P is ...

type Variant (Diskriminante: Diskret := Vorbelegung) is record ...
type Zeiger is access Variant;

package Legal is new P (Gezeigt => Variant,
                        Zeiger  => Zeiger);

subtype Invariant        is Variant (Wert);
subtype Invariant_Zeiger is Zeiger  (Wert);

package Illegal is new P (Gezeigt => Invariant,
                          Zeiger  => Invariant_Zeiger);
```

Ein Compiler weist die Deklaration des Pakets `Illegal` zurück mit Hinweis auf RM_95 12.5.4(3), ein anderer akzeptiert sie.

RM_95 12.5.4(3) verlangt, dass die bezeichneten Subtypen des aktuellen und des formalen Zugriffstyps statisch zueinander passen müssen. (*"For a formal access-to-object type, the designated subtypes of the formal and actual types shall statically match."*)

Passen sie nun oder passen sie nicht?