Gem #1: Лимитированные типы в Ада 2005 – лимитированные агрегаты

Автор: Bob Duff, AdaCore

Краткое содержание: Gem Ада #1 – При помощи Ада 2005 можно создавать лимитированные объекты с помощью агрегатов, что позволяет применять правила полного покрытия; ранее подобное было доступно только для нелимитированных типов.

Давайте начнем…

Одна из моих любимых особенностей Ada “правила полного покрытия” (“full coverage rules”) для агрегатов. Например, предположим, что у нас есть запись типа:

```
type Person is
   record
      Name : Unbounded_String;
      Age : Years;
   end record;
```

Мы можем создать объект типа с использованием агрегата:

```
X : constant Person :=
   (Name => To_Unbounded_String (”John Doe”),
    Age => 25);
```

Правила полного покрытия говорят, что каждый компонент типа `Person` должен учитываться в агрегате. Если позже мы изменим тип `Person`, добавив компонент `Shoe_Size`:

```
type Person is
   record
      Name : Unbounded_String;
      Age : Natural;
      Shoe_Size : Positive;
   end record;
```

и если мы забываем изменять X соответственно, компилятор напомнит нам, что необходимо назначить `Shoe_Size` значение типа `Positive`. Для оператора выбора `Case` также необходимо применять правила полного покрытия, которые служат той же цели.

Конечно, мы можем удовлетворить требования правил полного покрытия, используя **`others`** (обычно для агрегатов массивов и операторов выбора `Case`, но иногда полезно и для агрегатов записей). Например:

```
X : constant Person :=
   (Name => To_Unbounded_String (”John Doe”),
    others => 25);
```

По словам Ada RM **`others`** здесь означает именно то же самое значение, что для `Age` или `Shoe_Size`”. Но это неправильно: то, что **`others`** действительно означает “все

остальные компоненты, в том числе и те, что мы могли бы добавить на следующей неделе или в следующем году". Это означает, что Вы не должны использовать **others**, если вы уверены, что это должно распространяться на все случаи, которые еще не изобретены.

Пока что это старые новости – правила полного покрытия помогали в обслуживании с языком Ada 83. Итак, какое это имеет отношение к Ada 2005?

Предположим, у нас ограниченный тип (limited type):

```
type Limited_Person is limited
   record
      Self : Limited_Person_Access :=
                Limited_Person'Unchecked_Access;
      Name : Unbounded_String;
      Age : Natural;
      Shoe_Size : Positive;
   end record;
```

Этот тип имеет собственную ссылку; Нет смысла копировать такие объекты, потому что `Self` окажется в неправильном месте. Поэтому мы хотели бы сделать тип ограниченным – **limited**, чтобы программисты не могли случайно сделать копии. В конце концов, тип, вероятно, является частным, поэтому клиентский программист может не знать о проблеме. Мы также могли бы решить эту проблему с контролируемыми типами, но контролируемые типы являются дорогостоящими и, если не нужно, добавят излишнюю сложность.

В Ada 95 агрегаты были запрещены для ограниченных типов. Поэтому мы сталкиваемся с неэлегантным выбором решения: сделать тип ограниченным и инициализировать объект этого типа следующим образом:

```
X : Limited_Person;
X.Name := To_Unbounded_String ("John Doe");
X.Age := 25;
```

Такой выбор имеет проблему обслуживания, должны быть предотвращены все правила покрытия. Или, сделать тип неограниченным и получить преимущества от использования агрегатов, но потерять способность предотвращать копии.

В Ada 2005 агрегату позволено быть **limited**; мы можем например:

```
X : aliased Limited_Person :=
   (Self => null, – Wrong!

    Name => To_Unbounded_String (”John Doe”),
    Age => 25,
    Shoe_Size => 10);
X.Self := X'Access;
```

Мы увидим, что делать с этим «Self => null» в будущей Gem #2.

Следует отметить одно очень важное требование: реализация необходима для построения значения X «на месте»; Не допустимо построить агрегат во временной

переменной и затем скопировать его в X, потому что это нарушит весь смысл ограниченных объектов – вы не можете копировать их.