Gem #2: Лимитированные типы в Ada 2005 – Нотация типа <> в агрегатах

Автор: Bob Duff, AdaCore

Краткое содержание: Gem Ada #2 – Нотация типа <> может быть использована в агрегатах для запроса начального значения определенных компонентов.

Давайте начнем…

В Gem #1 мы отметили, что Ada 2005 разрешает агрегаты для ограниченных типов (**limited**). Такой агрегат должен использоваться для инициализации некоторого объекта (который включает в себя передачу параметров, где мы инициализируем формальный параметр). Агрегаты для лимитированных типов «встроены» в инициализируемый объект, то есть при передаче параметров в объект учитывается его окружение.

Вот пример из Gem #1:

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

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

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

Кажется неудобным устанавливать значение Self с неправильным значением (null), а затем исправлять его. Также кажется раздражающим, что у нас есть (правильное) значение по умолчанию для Self, но в Ada 95 мы не можем использовать значения по умолчанию с агрегатами. Ada 2005 добавляет новый синтаксис в агрегатах - «<>» означает «использовать значение по умолчанию, если есть».

Здесь, мы можем сказать:

```
X : aliased Limited_Person :=
   (Self => <>,
    Name => To_Unbounded_String ("John Doe"),
    Age => 25,
    Shoe_Size => 10);
```

«`Self => <>`» означает использовать значение по умолчанию `Limited_Person'Unchecked_Access`. Так как `Limited_Person` отображается внутри объявления типа, он ссылается на «текущий экземпляр» типа, который в данном случае является `X`. Таким образом, мы устанавливаем `X.Self` для `X'Unchecked_Access`.

Обратите внимание, что использование «<>» в агрегате может быть опасным, так как оно может оставить некоторые компоненты неинициализированными. «<>» означает «использовать значение по умолчанию». Если тип компонента скалярный и отсутствует компонент записи по умолчанию, то значение по умолчанию отсутствует.

Например, если у нас есть такой агрегат типа String:

```
Uninitialized_String_Const : constant String := (1..10 => <>);
```

Мы получим строку из 10 символов, все символы которой являются недопустимыми. Заметьте, что это не более и не менее опасно, чем это:

```
Uninitialized_String_Var : String (1..10); – no initialization

Uninitialized_String_Const : constant String := Uninitialized_String_Var;
```

Как всегда, нужно быть осторожным в отношении неинициализированных скалярных объектов.