Gem #12: Лимитированные типы в Ада 2005 - Нотация типа <> в агрегатах, часть 2

Автор: : Bob Duff, AdaCore

Краткое содержание: Gem Ада #12 - Данный пример демонстрирует, как с помощью нотации типа <> в агрегатах можно лучшим образом применять начальные значения компонента записи, чтобы избежать повторения кода.

Давайте начнем…

Вы когда-либо писали код Ada 95 как этот?

```
package P is
   type T is private;
   …
private
   type T is
      record
         Color : Color_Enum := Red;
         Is_Gnarly : Boolean := False;
         Count : Natural;
      end record;
end P;

package body P is
   Object_100 : constant T :=
      (Color => Red, Is_Gnarly => False, Count => 100);
   …
end P;
```

Мы хотим, чтобы Object_100 был инициализированным по умолчанию T, с Count равным 100. Это немного раздражает, что нам пришлось дважды писать значения по умолчанию Red и False. Что, если мы передумаем о Red и забудем изменить его во всех соответствующих местах?

Обозначение «<>» приходит на помощь. Если мы хотим сказать: «Сделайте Count равным 100, но инициализируйте Color и Is_Gnarly их значения по умолчанию», мы можем сделать это:

```
Object_100 : constant T :=
   (Color => <>, Is_Gnarly => <>, Count => 100);
```

С другой стороны, если мы хотим сказать: «Сделайте Count равным 100, но инициализируйте все остальные компоненты, включая те, которые мы можем добавить на следующей неделе, к их значениям по умолчанию», мы можем сделать это:

```
Object_100 : constant T := (Count => 100, others => <>);
```

Обратите внимание на то, что, если мы добавляем компонент “Glorp: Integer”; к типу T тогда случай “ **others** ” оставляет Glorp неопределенным, как этот код Ada 95 сделал бы:

```
Object_100 : T;
Object_100.Count := 100;
```

Думайте дважды перед использованием “ **others** ”.