Gem #10: Лимитированные типы в Ада 2005 - Расширенные операторы возврата

Автор: : Bob Duff, AdaCore

Краткое содержание: Gem Ада #10 - Для присвоения создаваемому в функцией объекту имени можно использовать в функции extended_return_statement.

Давайте начнем…

Общей идиомой в Ada 95 является создание результата функции в локальном объекте, а затем возврат этого объекта:

```
function Sum (A : Array_Of_Natural) return Natural is
   Result : Natural := 0;
begin
   for Index in A'Range loop
      Result := Result + A (Index);
   end loop;
   return Result;
end Sum;
```

Ada 2005 позволяет обозначить оператор расширенного возврата, который позволяет объявить объект результата и вернуть его как часть одного оператора. Это выглядит так:

```
function Sum (A : Array_Of_Natural) return Natural is
begin
   return Result : Natural := 0 do
      for Index in A'Range loop
         Result := Result + A (Index);
      end loop;
   end return;
end Sum;
```

Оператор возврата здесь создает `Result`, инициализирует его к 0 и выполняется, код между " **do** " и " **end return** ". Когда " **end return** " достигнут, `Result` автоматически возвращается как результат функции `Sum`.

Для большинства типов, в этом нет ничего сложного — это просто синтаксический сахар. Но для ограниченных типов, этот синтаксис является практически необходимым:

```
function Make_Task (Val : Integer) return Task_Type is
   Result : Task_Type (Discriminant => Val * 3);
begin
   … -- some statements
   return Result; -- Illegal!
end Make_Task;
```

Оператор return здесь является незаконным, потому что Result является локальным для Make_Task, и его возвращение будет включать в себя копию, которая не имеет смысла (поэтому типы задач ограничены). В Ada 2005 мы можем писать конструкторские функции для типов задач:

```
function Make_Task (Val : Integer) return Task_Type is
begin
   return Result : Task_Type (Discriminant => Val * 3) do
      … -- some statements
```

```
   end return;
end Make_Task;
```

Если мы вызываем его как этом примере:

```
My_Task : Task_Type := Make_Task (Val => 42);
```

Результат создается “на месте” в My_Task. Результат временно считают локальным для Make_Task во время выполнения части кода “… -- *some statements* ”, но как только Make_Task возвращается, задача становится более глобальной. Результат и My_Task действительно - один и тот же объект.

При возврате задачи из функции она активируется после возвращения функции. В части кода «… -- *some statements* ”,» лучше не пытаться вызвать одну из `entry` задачи, потому что это будет тупик. То есть, вызов `entry` будет ждать, пока задача не примет оператор accept, чего никогда не произойдет, потому что задача никогда не будет активирована.

Хотя extended_return_statement был добавлен в язык специально для поддержки ограниченных функций конструктора, он пригодится всякий раз, когда вам нужно локальное имя для результата функции:

```
function Make_String (…) return String is
   Length : Natural := 10;
begin
   if … then
      Length := 12;
   end if;
   return Result : String (1..Length) do
      … -- fill in the characters
      pragma Assert (Is_Good (Result)); null;
   end return;
end Make_String;
```

Обсуждение…