Gem #17: Прагма No_Return, Часть 2 (функции)

Автор: Bob Duff, AdaCore

Краткое содержание: Gem Ада #17 - GNAT выдает предупреждение при обнаружении функций, которые могут завершить выполнение без оператора Return, и иногда подобные предупреждения о неисправности ложные. Прагма No_Return позволяет избежать подобных ложных предупреждений.

Давайте начнем…

Функция всегда должна возвращать значение или вызывать исключение. То есть достижение «**end**» функции является ошибкой. В Ada эта ошибка обнаруживается двумя способами:

1. Каждая функция должна содержать хотя бы один оператор возврата. Это правило слишком либерально, потому что некоторый путь потока управления может обойти оператор return. Это правило слишком консервативно, потому что оно требует возврата, даже когда все, что вы хотите сделать, - это сделать исключение:

```
function F (...) return ... is
begin
   raise Not_Yet_Implemented;  --  Illegal!
end F;
```

Функция F не НЕПРАВИЛЬНАЯ — она просто не закончена. По-моему, это правило должно было быть отменено и заменено анализом потока управления, подобным тому, что делает GNAT (см. ниже).

2. Если функция достигает своего «конца», появляется Program_Error. Функция во время выполнения улавливает все такие ошибки, но это неудовлетворительно; проверка и выявление этой ситуации во время компиляции будет лучше.

Действительно, GNAT обеспечивает эффективную проверку во время компиляции: компилятор дает предупреждение, если есть какой-либо путь потока управления, который мог бы достигнуть оператора “ **end** ”:

```
function F (...) return ... is
begin
   if Some_Condition then
      return Something;
   elsif Some_Other_Condition then
      return Something_Else;
   end if;
   --  CAN get here!
end F;
```

Существует ошибка, если Some_Condition и Some_Other_Condition не охватывают все случаи, и GNAT будет предупреждать. Мы можем исправить это несколькими способами, такими как:

```
function F (...) return ... is
begin
   if Some_Condition then
```

```
         return Something;
      elsif Some_Other_Condition then
         return Something_Else;
      else
         raise Some_Exception;
      end if;
   end F;
```

Или:

```
   function F (...) return ... is
   begin
      if Some_Condition then
         return Something;
      else
         pragma Assert (Some_Other_Condition);
         return Something_Else;
      end if;
   end F;
```

или (и вот **pragma** No_Return):

```
procedure Log_Error (...);
pragma No_Return (Log_Error);

function F (...) return ... is
begin
   if Some_Condition then
      return Something;
   elsif Some_Other_Condition then
      return Something_Else;
   else
      Log_Error (...);
   end if;
end F;
```

GNAT не поднимет ложную тревогу на последних трех версиях F, потому что GNAT делает простой анализ потока управления, чтобы доказать, что каждый путь достигает или оператора возврата, или оператора повышения или процедуры невозврата (который, по-видимому, содержит оператор raise, прямо или косвенно).

Один из рассмотренных способов состоит в том, что **pragma** No_Return позволяет Вам обертывать оператор **raise** в немного высокоуровневую абстракцию, не теряя вышеупомянутый анализ потока управления. No_Return - часть контракта; F полагается на Log_Error, не возвращаясь с оператором **return**, и компилятор гарантирует, что тело Log_Error повинуется этому контракту.

Обсуждение…