Gem #150: Вышедшие и неинициализированные

Автор: Robert Dewar — AdaCore

Краткое содержание: В данном Gem описываются некоторые, возможно, неожиданные случаи, когда переменные не всегда корректирует свое значение после присваивания, что нельзя сразу понять из кода

Давайте начнём…

Возможно, удивительно, что стандарт Ada указывает случаи, когда объекты, переданные в Out и In Out параметры, могут не обновляться, когда процедура завершается из-за исключения. Давайте возьмем пример:

```
with Ada.Text_IO;  use Ada.Text_IO;
procedure Gem is

   procedure Local (A : in out Integer; Error : Boolean) is
   begin
      A := 1;

      if Error then
         raise Program_Error;
      end if;
   end Local;

   B : Integer := 0;

begin
   Local (B, Error => True);
exception
   when Program_Error =>
      Put_Line ("Value for B is" & Integer'Image (B));  --  "0"
end Gem;
```

Эта программа выводит значение `0` для `B`, тогда как код указывает, что `A` назначается до возбуждения исключения, и поэтому читатель может ожидать, что `B` также будет обновлен.

Однако подвох заключается в том, что компилятор должен по умолчанию передавать объекты элементарных типов (скаляры и типы доступа) путем копирования и может решить сделать это для других типов (например, для записей), в том числе при передаче и выходе параметров. Так что происходит, когда формальный параметр `A` правильно инициализирован, исключение возникает до того, как новое значение `A` будет скопировано обратно в `B` (копирование произойдет только при нормальном возврате).

В общем, любой код, который читает фактический объект, переданный в параметр `out` или `in out` после исключения, является подозрительным и его следует избегать. GNAT имеет здесь полезные предупреждения, так что если мы упростим приведенный выше код до:

```
with Ada.Text_IO;  use Ada.Text_IO;
procedure Gem2 is

    procedure Local (A : in out Integer) is
    begin
       A := 1;
```

```
      raise Program_Error;
   end Local;

   B : Integer := 0;

begin
   Local (B);
exception
   when others =>
      Put_Line ("Value for B is" & Integer'Image (B));
end Gem2;
```

Теперь мы получаем предупреждение компиляции:

```
gem.adb:6:10: warning: assignment to pass-by-copy formal may have no effect
gem.adb:6:10: warning: "raise" statement may result in abnormal return (RM 6.4.1(17))
```

Разумеется, GNAT не может указать на все такие ошибки (см. Первый пример выше), что в целом потребовало бы полного анализа потока.

Поведение отличается при использовании типов параметров, которые стандарт обязывает передавать по ссылке, например, таких как теговые типы. Поэтому следующий код будет работать как положено, обновляя фактический параметр, несмотря на исключение:

```
procedure Gem3 is

   type Rec is tagged record
      Field : Integer;
   end record;

   procedure Local (A : in out Rec) is
   begin
      A.Field := 1;
      raise Program_Error;
   end Local;

   V : Rec;

begin
   V.Field := 0;
   Local (V);
exception
   when others => Put_Line ("Value of Field is" & V.Field'Img); -- "1"
end Gem3;
```

Стоит упомянуть, что GNAT обеспечивает прагму по имени `Export_Procedure`, который вызывает ссылочную семантику на параметрах. Использование этой прагмы гарантировало бы обновления фактического параметра до аварийного завершения процедуры. Однако эта прагма только относится к процедурам уровня библиотеки, таким образом, примеры выше должны быть переписаны для предотвращения использования вложенной процедуры, и действительно эта прагма предназначается главным образом для использования во взаимодействии через интерфейс с внешним кодом. Код ниже показывает пример, который гарантирует, что `B` установлен в `1` после вызова к `Local`:

```
package Gem4_Support is
```

```
  procedure Local (A : in out Integer; Error : Boolean);
  pragma Export_Procedure (Local, Mechanism => (A => Reference));

end Gem4_Support;


package body Gem4_Support is

   procedure Local (A : in out Integer; Error : Boolean) is
   begin A := 1;
      if Error then
         raise Program_Error;
      end if;
   end Local;

end Gem4_Support;


with Ada.Text_IO;  use Ada.Text_IO;
with Gem4_Support; use Gem4_Support;
procedure Gem4 is
   B : Integer := 0;
begin
   Local (B, Error => True);
exception
   when Program_Error =>
      Put_Line ("Value for B is" & Integer'Image (B)); -- "1"
end Gem4;
```

В случае прямого присваивания глобальным переменным поведение при наличии исключений несколько отличается. Для предопределенных исключений, прежде всего Constraint_Error, разрешения на оптимизацию предоставляют некоторую гибкость в отношении того, обновляется ли глобальная переменная или нет, когда возникает исключение (см. Ada RM 11.6). Например, следующий код делает неверное предположение:

```
X := 0;     -- about to try addition
Y := Y + 1; -- see if addition raises exception
X := 1      -- addition succeeded
```

Программа не оправдана в предположении, что X = 0, если добавление вызывает исключение (предполагая, что X является глобальным здесь). Таким образом, любые такие предположения в программе являются неправильным кодом, который должен быть исправлен.

Связанный со статьёй текст программы

Файлы примеров Ada Gems распространяются AdaCore и могут быть использованы или изменены для любых целей без ограничений.

Об авторе

Автор: Robert Dewar — AdaCore

Д-р Роберт Дьюар (Robert Berriedale Keith Dewar, June 21, 1945 – June 30, 2015) является (*прим.* на момент публикации статьи) соучредителем, президентом и генеральным директором AdaCore и почетным профессором информатики в Нью-Йоркском университете. С акцентом на разработку и внедрение языка программирования, д-р Дьюар был основным вкладчиком в Ada на протяжении всей его эволюции и является главным архитектором технологии AdaCore GNAT Ada. Он был соавтором компиляторов для SPITBOL (SNOBOL), Realia COBOL для ПК (теперь продается Computer Associates) и Alsys Ada, а также написал несколько операционных систем в реальном времени для Honeywell Inc. Доктор Дьюар выступал с докладами и докладами по вопросам языка программирования и сертификации безопасности, и, как эксперт по компьютерам и законодательству, он часто приглашается на конференции, чтобы выступить по вопросам программного обеспечения с открытым исходным кодом, лицензирования и смежным темам.

См. также на https://en.wikipedia.org/wiki/Robert_Dewar

Last Updated: 10/13/2017
Posted on: 6/17/2013

Обсуждение…