Gem #31: Предусловия/постусловия

Автор: Robert Dewar, AdaCore

Краткое содержание: Gem Ада #31 - Понятие предусловий и постусловий определено уже давно. Предусловие - это такое условие, которое должно быть соблюдено (возвращать значение true) перед тем, как часть программы начнёт своё исполнение. Постусловие, в свою очередь, - условие, которое должно быть соблюдено (возвращать значение true) после того, как часть программы закончила свое исполнение.

Давайте начнем…

Понятие предварительных условий и постусловий определено уже давно. Предварительным условием является условие, которое должно быть истинным до того, как будет выполняться раздел кода, и постусловие является условием, которое должно быть истинным после выполнения раздела кода.

В контексте, о котором мы говорим здесь, раздел кода всегда будет подпрограммой. Предварительные условия - это условия, которые должны быть гарантированы вызывающим абонентом перед вызовом, а постусловия - результатом, гарантированным самим кодом подпрограммы.

Возможно, используя pragma Assert (как определено в Ada 2005 и как реализовано во всех версиях GNAT), для аппроксимации проверок времени выполнения, соответствующих предварительным условиям и постусловиям, путем размещения утвердительных прагм в теле подпрограммы, но есть несколько проблемы с этим подходом:

1. Утверждения не видны в спецификации, а предпосылки и постусловия логически являются частью (на самом деле, важной частью) спецификации.
2. Постусловия должны повторяться в каждой точке выхода.
3. Постусловия часто ссылаются на исходное значение параметра на входе или результат функции, и нет простого способа сделать это в утверждении.

В последних версиях GNAT реализованы две прагмы: предварительное условие (Precondition) и постусловия (Postcondition), которые позволяют решить все три проблемы удобным способом. Самый простой способ описать этот подход - использовать пример:

```
package Arith is
   function Sqrt (Arg : Integer) return Integer;
   pragma Precondition (Arg >= 0);
   pragma Postcondition
    (Sqrt'Result >= 0
       and then
    (Sqrt'Result ** 2) <= Arg
       and then
    (Sqrt'Result + 1) ** 2 > Arg);
end Arith;

with Arith; use Arith;
with Text_IO; use Text_IO;
procedure Main is
begin
   Put_Line (Sqrt (9)'Img);
```

```
      Put_Line (Sqrt (10)'Img);
      Put_Line (Sqrt (-3)'Img);
   end;
```

Теперь, если мы скомпилируем с ключом компиляции -gnata (который разрешает предусловия и постусловия), то в результате у нас есть корректное тело для Sqrt, и при запуске Main мы получим:

```
 3
 3

raised SYSTEM.ASSERTIONS.ASSERT_FAILURE :
   Precondition failed at arith.ads:3
```

Теперь, если с телом Sqrt было что-то не так, что дало неправильный ответ, мы могли бы получить:

```
raised SYSTEM.ASSERTIONS.ASSERT_FAILURE :
  postcondition failed at arith.ads:4
```

Указывая, что у нас есть ошибка в теле Sqrt, которую мы должны исследовать.

Следует еще упомянуть, что это обещанная способность ссылаться на старое значение параметров. Новый атрибут `Old позволяет это, как показано в этом примере:

```
procedure Write_Data (Total_Writes : in out Natural);
--  Write out the data incrementing Total_Writes to
--  show number of write operations performed.
pragma Postcondition (Total_Writes > Total_Writes'Old);
```

Введение предварительных условий и постусловий в GNAT является мощным инструментом для проектирования и документации (Eiffel назвал этот подход «дизайном по контракту»).

Предусловия и постусловия выполняют три функции

1. Они обеспечивают ценную официальную документацию в спецификации
2. Они снабжают входной сигнал инструменты доказательства
3. Они помогают найти ошибки в ходе обычной отладки, как показано на примере выше.

Поддержка предварительных условий и постусловий будет доступна в GNAT Pro 6.2.1 и в следующих версиях GNAT GPL. Если вы являетесь пользователем GNAT Pro и хотите попробовать эту функцию сегодня, запросите wavefront с помощью Gnat Tracker.

Связанный исходный код

Attached Files

```
package pkg is
   function Sqrt (Arg : Integer) return Integer;
   pragma Precondition (Arg >= 0);
   pragma Postcondition
```

```
      (Sqrt'Result >= 0
       and then
         (Sqrt'Result ** 2) <= Arg
       and then
         (Sqrt'Result + 1) ** 2 > Arg);
end pkg;

with pkg; use pkg;
with Text_IO; use Text_IO;
procedure Main is
begin
   Put_Line (Sqrt (9)'Img);
   Put_Line (Sqrt (10)'Img);
   Put_Line (Sqrt (-3)'Img);
end;
package body pkg is
   function Sqrt (Arg : Integer) return Integer is
      Result : Integer := 0;
   begin
      loop
         Result := Result + 1;
         if Result ** 2 > Arg then
            return Result;
         end if;
      end loop;
   end;
end pkg;
```

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

Об авторе

Robert Dewar (1945-2015)
http://www.adacore.com/press/adacore-president-robert-dewar-1945-2015/
AdaCore

Д-р Роберт Дьюар являлся соучредителем, президентом и генеральным директором AdaCore и заслуженным профессором компьютерных наук в Нью-Йоркском университете. Сфокусировавшись на разработке и внедрении языка программирования, д-р Дьюар был основным вкладчиком в Ada на протяжении всей своей эволюции и являлся главным разработчиком технологии AdaCore GNAT Ada. Он совместно разработал компиляторы для SPITBOL (SNOBOL), Realia COBOL для ПК (в настоящее время продается Computer Associates) и Alsys Ada, а также написал несколько операционных систем реального времени для Honeywell Inc. Д-р Дьюар поставлял документы и презентации по вопросам языка программирования и сертификации по безопасности, а также как эксперт по компьютерам и законодательству, его часто приглашали на конференции, чтобы поговорить о программном обеспечении с открытым исходным кодом, вопросах лицензирования и смежных вопросах.

Last Updated: 10/13/2017
Posted on: 4/14/2008

Обсуждение…

1. 16 апреля 2008 года в 3:32

Джефф Крем сказал:

Хорошее и интересное сейчас время. Я просто перечитывал некоторые из старых AIs в этой области и думал, что хочу, чтобы это было реализовано в Ada 2005.

Одна из проблем, которые выражались в AIs, заключалась в том, что в Ada мы обычно хотим проверять во время статической компиляции, а в Gem-031 рассматривалось только как динамическая проверка, но, надеюсь, такие инструменты, как Polyspace и другие статические инструменты, могут начать использовать эту информацию. Даже не говоря уже о том, что мне нравится идея использовать это, чтобы помочь документировать ожидания.

2. 16 апреля 2008 года в 19:17

Кирилл Комар сказал:

Спасибо за хороший комментарий Джеффа. Вопрос о статических и динамических условиях pre / post действительно очень интересен. Инструменты статического анализа, использующие информацию, предоставляемую динамическими версиями, уже на пути ... Подробнее об этом позже ;-)

Статические версии уже были предоставлены такими инструментами, как Spark от Praxis.

Найти способ объединить статические и динамические версии - интересная задача, которую мы серьезно рассматриваем!