Gem #63: Действие прагмы Suppress.

Автор: Gary Dismukes, AdaCore

Краткое содержание: Gem Ada #63 - Функции Ada в основном нацелены на предотвращение нарушений свойств типов данных, что достигается либо ограничениями на стадии компилирования, либо, в случае динамических свойств, - проверками в ходе выполнения. Тем не менее, проверки в ходе выполнения в Ada можно отменить, но разработчику не стоит пользоваться этой функцией с целью обхода системы контроля типов данных.

Давайте начнём…

Одной из ключевых сильных сторон Ada всегда была ее сильная типизация. Язык накладывает строгую проверку свойств типа и подтипа, чтобы помочь предотвратить случайные нарушения системы типов, которые являются общим источником программных ошибок в других менее строгих языках, таких как C. это делается с использованием комбинации ограничений времени компиляции (правила законности), которые запрещают смешивание значений различных типов, вместе с проверками времени выполнения, чтобы поймать нарушения различных динамических свойств. Примеры проверяют значения против ограничений подтипа и предотвращают разыменования значений доступа null.

В то же время Ada предоставляет определенные "лазейки", такие как Unchecked_Conversion, которые позволяют выборочно обходить обычные функции безопасности, что иногда необходимо при взаимодействии с оборудованием или кодом, написанным на других языках.

Ada также позволяет явное подавление проверок во время выполнения, которые существуют, чтобы гарантировать, что различные свойства объектов не нарушены. Это подавление может быть сделано с помощью Pragma Suppress, а также с помощью переключателя времени компиляции на большинстве реализаций (в случае GNAT, c-gnatp коммутатор).

Помимо разрешения подавления всех проверок, Pragma Suppress поддерживает подавление определенных форм проверки, таких как Index_Check для индексирования массива, Range_Check для проверки скалярных границ и Access_Check для разыменования значений доступа. (См. Раздел 11.5 справочного руководства Ada для получения дополнительной информации.)

Вот простой пример подавления проверок индекса в рамках определённой подпрограммы:

```
procedure Main is
   procedure Sort_Array (A : in out Some_Array) is
      pragma Suppress (Index_Check);  -- eliminate check overhead
   begin
      ...
   end Sort_Array;
end Main;
```

Однако в отличие от такой функции, как Unchecked_Conversion, целью подавления проверки не является включение программ для подрыва системы типов, хотя многие программисты, похоже, имеют это неправильное представление.

Что важно понимать в pragma Suppress, так это то, что она только дает разрешение на реализацию для удаления проверок, но не требует такого исключения. Намерение подавить не позволить обойти семантику Ada, а скорее улучшить эффективность, и Справочное руководство Ada имеет четкое заявление на этот счет в примечании в RM-11.5, пункт 29:

There is no guarantee that a suppressed check is actually removed; hence a pragma Suppress should be used only for efficiency reasons.

Нет никакой гарантии, что погашенная проверка фактически удалена; следовательно Pragma Suppress должна использоваться только по причинам эффективности.

Существует связанный с политикой реализации компилятора Ada, Совет по Внедрению, который рекомендует, чтобы реализации должны сводить к минимуму код, выполняемый для проверок, которые были подавлены, но программист по-прежнему несёт ответственность за то, чтобы правильное функционирование программы не зависело от не выполняемых проверок.

Существуют разные причины, по которым компилятор может отказаться от удаления проверки. На некоторых аппаратных средствах некоторые проверки могут быть по существу бесплатными, например, проверка нулевого указателя или арифметическое переполнение, и это может оказаться непрактичным или добавить дополнительные затраты для подавления проверки. Другим примером, где было бы нецелесообразно удалять проверки, является операция, выполняемая вызовом во время выполнения, где проверка может быть лишь малой частью более дорогостоящей операции, выполненной дополнительно в потоке выполняемых операций.

Кроме того, во многих случаях GNAT может определить во время компиляции, что определённая проверка времени выполнения будет нарушена. В таких ситуациях он даёт предупреждение о том, что исключение будет поднято, и генерирует код специально для повышения исключения. Вот пример:

```
X : Integer range 1..10 := ...;

..

if A > B then
   X := X + 1;
   ..
end if;
```

Для назначения приращения X компилятор обычно генерирует машинный код, эквивалентный:

```
Temp := X + 1;
if Temp > 10 then
   raise Constraint_Error;
end if;
X := Temp;
```

Если ограничения диапазона подавляются, то компилятор может просто генерировать приращение и присвоение. Однако, если компилятор способен каким-то образом доказать, что X = 10 в этот момент, он выдаст предупреждение и просто заменит всё присваивание:

```
raise Constraint_Error;
```

хотя проверки подавляются. Это уместно, потому что (1) мы не заботимся об эффективности кода заведомо ошибочного кода, и (2) нет никаких "дополнительных" затрат на проверку, потому что, если мы достигнем этой точки, код безоговорочно потерпит неудачу в выполнении.

Еще одна важная вещь, которую следует отметить о проверках и Pragma Suppress, - это утверждение в ADA RM (RM-11.5, пункт 26):

If a given check has been suppressed, and the corresponding error situation occurs, the execution of the program is erroneous.

Если данная проверка была подавлена, и возникает соответствующая ситуация ошибки, выполнение программы является ошибочным.

В Ada ошибочное выполнение-это плохая ситуация, потому что это означает, что выполнение вашей программы может иметь произвольные неприятные эффекты, такие как непреднамеренная перезапись памяти. Обратите внимание, что программа, "правильное" выполнение которой каким-либо образом зависит от подавляемой проверки, может работать так, как ожидает программист, но все равно может завершиться ошибкой при компиляции с другим компилятором или для другой цели, или даже с более новой версией того же компилятора. Другие изменения, такие как включение оптимизации или внесение изменений в совершенно несвязанную часть кода, также могут привести к сбою кода.

Так что это точно не умно писать код, который зависит от проверки их удалить. На самом деле, действительно имеет смысл подавлять проверки только тогда, когда есть веские основания полагать, что проверки не могут потерпеть неудачу в результате тестирования или другого анализа. В противном случае вы удаляете важную функцию безопасности Ada, которая призвана помочь поймать ошибки.

Связанный со статьёй текст программы

Attached Files отсутствуют

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

Об авторе

Гари Дисмукес (Gary Dismukes) начал свое участие в Ada в 1980 году, начав разработку компилятора Ada, работая под руководством доктора Кеннета Боулза (Dr. Kenneth Bowles) в проекте UCSD Pascal, до окончания Калифорнийского университета в Сан-Диего (бакалавр математики, бакалавр и магистр компьютерных наук). Эта работа перешла на TeleSoft, один из ранних поставщиков Ada, где он работал в качестве главного инженера-программиста с 1981 по 1994 год. Гари присоединился к AdaCore в 1994 году. В дополнение к участию в разработке gnat front end, он также работал над реализацией продукта GNAT-for-Java и серверной части, ориентированной на проприетарное семейство процессоров на основе стека для одного из крупнейших клиентов AdaCore. Он также участвовал в качестве видного рецензента и члена группы докладчиков Ada (ARG) по Ada 95 и продолжает работать в качестве члена ARG Ada 2005. В дополнение к своему энтузиазму к Аде, Гари является поклонником колдовских искусств и предоставил магические развлечения на нескольких функциях компании AdaCore.

Last Updated: 10/13/2017
Posted on: 4/21/2009

Обсуждение…

2 ответа на " Gem #63: эффект Pragma Suppress”

1. 13 мая 2009 года в 15: 59

Питер Германн (Peter Hermann) сказал:

Текст умело описывает философию безопасности в Ada.

Для новичков должен быть хотя бы один пример

pragma suppress (на Ваше усмотрение) практического применения.

2. 18 мая 2009 года в 8: 53

Гари Дисмукес (Gary Dismukes) сказал:

Спасибо за ваши комментарии Питер. Следуя вашему предложению, был добавлен простой пример использования Pragma Suppress.