Gem #156: Контроль отображения в GNAT

Автор: Robert Dewar - AdaCore

Краткое содержание: В компиляторе существует много вариантов генерации и контроля выходных данных. Они часто малоизвестны, так как информация о них спрятана в кипах сложной документации. В данном Gem будут продемонстрированы данные варианты и то, как их можно использовать для контроля исходящих данных из компилятора.

Давайте начнём…

Вывод по умолчанию от компилятора включает в себя только сообщения об ошибках, а также все предупреждения, которые включены по умолчанию.

Например:

```
f.adb:3:04: warning: "return" statement missing following this statement
f.adb:3:04: warning: Program_Error may be raised at run time
f.adb:4:14: warning: value not in range of type "Standard.Natural"
f.adb:4:14: warning: "Constraint_Error" will be raised at run time
f.adb:6:16: division by zero
f.adb:6:16: static expression fails Constraint_Check
```

Эти сообщения показывают точное местоположение сообщений, и если вы отредактируете файл, вы сможете узнать, где именно отправлено каждое сообщение. Но есть много переключателей, которые можно использовать для изменения вывода. Чтобы лучше видеть, где выдается каждое сообщение, не генерируя слишком много выходных данных, вы можете использовать -gnatv:

Эти сообщения показывают точное расположение сообщений, и при редактировании файла можно точно узнать, где каждое сообщение выдается. Но есть много переключателей,которые можно использовать для изменения вывода. Чтобы лучше видеть, где каждое сообщение выдается, не создавая слишком много выходных данных, вы можете использовать `-gnatv`:

```
3.    if A > B then
      |
   >>> warning: "return" statement missing following this statement
   >>> warning: Program_Error may be raised at run time
4.       return -1;
                |
   >>> warning: value not in range of type "Standard.Natural"
   >>> warning: "Constraint_Error" will be raised at run time
6.       return 5 / 0;
                  |
   >>> division by zero
   >>> static expression fails Constraint_Check
```

И если вы используете `-gnatl`, вы можете получить полный список с номерами строк и всеми сообщениями:

```
Compiling: f.adb (source file time stamp: 2013-12-28 18:26:22)

     1. function F (A, B : Natural) return Natural is
     2. begin
     3.    if A > B then
           |
        >>> warning: "return" statement missing following this statement
```

```
   >>> warning: Program_Error may be raised at run time
4.       return -1;
                |
   >>> warning: value not in range of type "Standard.Natural"
   >>> warning: "Constraint_Error" will be raised at run time
5.    elsif B = 0 then
6.       return 5 / 0;
                  |
   >>> division by zero
   >>> static expression fails Constraint_Check
7.    end if;
8. end F;
```

Обратите внимание, что в приведенном выше выводе временная метка исходного файла может раздражать, если, например, вы не выводите тест регрессии, но ее можно подавить с помощью ключа `-gnatd7`. Также-обычно принимает необязательный параметр (например,`-gal=f.list`), который позволяет записывать этот вывод в указанный файл.

В приведенном выше выводе у нас есть сообщения, которые распространяются на две строки. Ключ `-gnatjnn`, где `nn` - десятичное целое число, обеспечивает хороший способ вывода таких сообщений. Значение `nn` - это максимальная длина строки, поэтому, например, если мы хотим ограничить длину выходного сообщения 68 символами, мы можем использовать ключи `-gnatl` и-`gnatj68`:

```
1. function F (A, B : Natural) return Natural is
2. begin
3.    if A > B then
      |
   >>> warning: "return" statement missing following this
       statement, Program_Error may be raised at run time
4.       return -1;
                |
   >>> warning: value not in range of type "Standard.Natural",
       "Constraint_Error" will be raised at run time
5.    elsif B = 0 then
6.       return 5 / 0;
                  |
   >>> division by zero, static expression fails
       Constraint_Check
7.    end if;
8. end f;
```

Ключ `-gnatj` - довольно недавнее дополнение, и многие люди не знают об этом, но это определенно приятно во многих ситуациях.

В дополнение к основному управлению выходом источника, различные вспомогательные выходы, которые полезны. Особый интерес представляет `-gnatR` , заставляя компилятор печатать о представлении информации, включая размеры и выравнивания блоков памяти, что может быть очень полезно для диагностики проблем при взаимодействии с внешними системами и оборудованием.

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: 2/4/2014

Обсуждение…