Gem #91: GPS - автоматическое дополнение ввода (Часть 2 из 2)

Автор: Quentin Ochem, AdaCore

Краткое содержание: В Gem #91 продолжается тема автоматического дополнения ввода GPS, начатая в первой части этой серии публикаций. В Gem #91 можно ознакомиться с новшествами, которые станут доступны в следующей версии GPS.

Давайте начнём…

В этом Gem мы собираемся заполнить дополнительные части исходных файлов Ada, созданных в первой части этой серии (Gem #88), с использованием механизма интеллектуального завершения GPS. Вся семантическая информация, необходимая для обеспечения этого вычисления, выполняется «на лету». Обратите внимание, что функции, описанные в этом Gem, пока недоступны, и будут выпущены в качестве части версии GPS после 4.4.0. Пользователи, поддерживаемые GNAT Pro, могут запросить ранний доступ к версии GPS с этими возможностями.

Механизм завершения основывается на парсере, запущенном при запуске GPS. Прогресс парсера появится в правом нижнем углу окна GPS. Имейте в виду, что завершение может быть неточным до завершения анализа.

Заполнение агрегатов

GPS может автоматически заполнить квалифицированные агрегаты. Вместо того, чтобы писать серию присвоений отдельным компонентам переменной, попробуем автоматически инициализировать значение в целом. Введите `"A_Variable := Rec'("` Левая скобка является автоматическим триггером для механизма завершения. Первая запись предлагает выбор заполнения всех полей для этой записи в названном агрегате - только значения компонентов должны быть предоставляемые пользователем. Другие записи предлагают выбор добавления нового имени в список значений для совокупности.

Завершение прагм и атрибутов

Предположим, мы хотим добавить утверждение перед вызовом `Arctan`, проверяя, что аргумент больше или равен нулю.

Перед заявлением, содержащим вызов, введите «`pragma`». Функция интеллектуального завершения будет запускаться автоматически. Отображается список всех доступных `pragma` с соответствующей документацией. Прокрутите вниз до пункта `Assert`. Теперь у вас есть доступ к документации для `pragma Assert`. Нажмите кнопку ввода, как только вы закончите просмотр документации, и заполните её, например: «`pragma Assert (X> = 0)`».

Атрибуты могут быть перечислены одинаково. Например, при вводе `X'` запускается интеллектуальное завершение, и перечислены все доступные атрибуты.

Завершением родовой сущности объектов

Рассмотрим пример контейнера Ada в нашем приложении. Начните с простой основной подпрограммы:

```
with Ada.Containers.Doubly_Linked_Lists;
use Ada.Containers;

package Main is
   type Rec is record
      A, B, C : Integer;
   end record;
```

```
begin
   null;
end Main;
```

Затем в объявлениях введите: "`package R_List is new Doubly_Linked_Lists (`". Дождитесь вывода подсказки автоматического интеллектуального завершения ввода, и вы можете ввести формальную часть. Добавьте просто `Rec` выбрав здесь в качестве фактического параметра `Element_Type`. Введите предложение `use R_List;`. Теперь вы должны иметь:

```
package R_List is new Ada.Containers.Doubly_Linked_Lists (Element_Type => Rec);
use R_List;
```

Объявите переменную этого типа списка, например «`L: R_List.List;`». Затем в последовательности операторов после `begin` добавьте несколько элементов, например "`L. Append (Rec'(0, 0, 0));`". Затем попробуйте получить доступ к первому элементу. Введите «`L.First_Element.`». Функция завершения понимает общий экземпляр и предлагает завершить выбранное имя одним из трёх полей: `A, B` или `C`.

На этом примере мы завершаем наш небольшой учебник по использованию возможности интеллектуального автоматического помошника завершения вводимой информации GPS. Как уже упоминалось, функции, представленные в этой части 2 Gem, будут доступны в предстоящем выпуске GPS (версия 5.0.0).

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

Quentin Ochem имеет опыт разработки программного обеспечения, специализирующийся на разработке программного обеспечения для критически важных приложений. Он имеет более чем 10-летний опыт разработки Ada. Сегодня он работает техническим менеджером AdaCore по проектам, связанным с авионикой, железнодорожной, космической и оборонной отраслями. Он также обучает стандарту DO-178B авионики в университете EPITA в Париже.

Last Updated: 10/13/2017
Posted on: 9/27/2010

Обсуждение…