Gem #88: GPS - автоматическое дополнение ввода (Часть 1 из 2)

Автор: Quentin Ochem, AdaCore

Краткое содержание: В данном Gem в виде легко воспроизводимого на практике руководства будет показано, как пользоваться автоматическим дополнением ввода. Здесь будут описаны уже доступные в GPS 4.4.0 возможности. В дальнейшем будут представлены новые возможности, реализованные в следующей версии GPS.

Давайте начнём…

В этом Gem мы собираемся создать несколько файлов, используя механизм завершения. Вся семантическая информация, необходимая для обеспечения этого вычисления, выполняется «на лету». Для того, чтобы следовать, просто откройте GPS по любому проекту.

Обратите внимание, что механизм завершения основывается на специальном парсере, запущенном при запуске GPS. Вы можете увидеть прогрессию анализатора в правом нижнем углу экрана. Завершение может быть неточным до завершения анализа.

Простое завершение компонентов

Создайте новый файл main.adb:

```
procedure Main is
begin
   null;
end Main;
```

Сначала мы объявим несколько типов и объектов. Создайте новый тип записи в декларативной части, а затем переменную этого типа, например:

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

A_Variable : Rec;
```

Затем в последовательности операторов введите «A» и нажмите <control-space>. <control-space> - это ярлык для ручного запроса завершения. Открывается всплывающее окно с указанием всех объявлений и ключевых слов, начинающихся с «A». На этом этапе их слишком много, поэтому давайте сократим список, добавив символ «_». Список теперь намного короче. Вы можете выбрать соответствующее завершение. A_Variable будет введен в текст.

<control-space> может использоваться в любое время для запроса завершения. При написании кода завершение может быть инициировано в определенных случаях. Введите в код точку ('.'). Всплывающее окно интеллектуального завершения автоматически запускается и предлагает выполнить компоненты Rec, а именно A и B.

Эта информация полностью синхронизируется с текущим содержимым редактора. Добавление компонента, например C, в представление завершения, а затем запрос на завершение снова, который покажет уже три компонента.

Завершение предложений with и use

Теперь мы добавим некоторые предложения with и use в программу. В частности, скажем, мы хотим добавить зависимость от какой-то стандартной единицы, которая обеспечивает математические операции.

Напишите "with" вверху основной подпрограммы. Запросите завершение, введя <control-space>. Все пакеты, которые можно "переключить", теперь перечислены здесь. Вы можете сузить список возможностей, написав, например, " Ad " и выбрав Ada. Добавить точку. Механизм завершения буду перечислять всех потомков определённого пакета Ада. Вы можете выбрать "Numerics". Интересно то, что на данном этапе мы можем не знать, что доступно в пакете Numerics. При добавлении ещё одной точки будут перечислены все дочерние пакеты Numerics. Прокрутив список вниз, увидим, что "Elementary_Functions" звучит как то, что нам нужно. Значит, выберем её.

Завершение профилей подпрограммы

Создайте новую переменную Float, такую как «X: Float;». Мы будем использовать эту переменную в математическом вычислении. Поскольку мы до сих пор не знаем, что находится в пакете «Elementary_Functions», мы начнем писать полностью префиксный вызов. Введите «X: = Ada.Numerics.Elementary_Functions.». Теперь вы можете увидеть все функции, перечисленные в всплывающем окне завершения. Выберите, например, Arctan и введите левую круглую скобку. Механизм завершения предлагает несколько профилей. Красные бриллианты обеспечивают полное завершение профиля с помощью обозначений. Выберите первый и укажите значение X и Y.

Завершение и ООП

Рассмотрим простой пакет с именем «Base»:

```
package Base is

   type Root is tagged private;

   procedure Prim (V : Root; I1, I2 : Integer);

   type Child is new Root with private;

   procedure Prim2 (V : Child);

private

   type Root is tagged record
      A : Integer;
   end record;

   type Child is new Root with record
      B : Integer;
   end record;

end Base;
```

В Main создайте две переменные V1 и V2 типа Root и Child:

```
   V1 : Root;
   V2 : Child;
```

При завершении, скажем, «V1.» Механизм завершения предлагает префиксные примитивы, такие как «Prim». При заполнении «V2.» Предлагаются неявно унаследованные примитивы «Prim» и недавно объявленные «Prim2».

Завершение также чувствительно к контексту видимости. Давайте создадим тело для Base:

```
package body Base is

   procedure Prim (V : Root; I1, I2 : Integer) is
   begin
```

```
      null;
   end Prim;

   procedure Prim2 (V : Child) is
   begin
      null;
   end Prim2;

end Base;
```

Попробуйте написать «V.» в теле Prim2. Теперь у вас будет доступ ко всем видимым компонентам Child, включая поля, которые скрыты от Main из-за приватной части пакета.

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: 6/14/2010

Обсуждение…