Gem #127: Итераторы в Ada 2012 - Часть 1.

Автор: Emmanuel Briot — AdaCore

Краткое содержание: Начало цикла статей посвященных новому стандарту Ada 2012. Итераторы в Ada 2012 облегчают восприятие текста структур данных. В данном Gem #127 описывается новый синтаксис и его взаимодействие с языком.

Давайте начнём…

В следующих примерах предполагается, что мы создали экземпляр списка Ada, например:

```
with Ada.Containers.Doubly_Linked_Lists;

...

declare
   package Integer_Lists is
      new Ada.Containers.Doubly_Linked_Lists (Integer);
   use Integer_Lists;

   L : Integer_Lists.List;

begin
   L.Append (10);
   L.Append (20);
end;
```

В Ada 2005 итерация по этому списку будет выглядеть так:

```
declare
   C : Integer_Lists.Cursor;
begin
   C := First (L);
   while Has_Element (C) loop
      --  Print current value
      Put_Line (Integer'Image (Element (C)));

      --  Change the element in place in the list
      Replace_Element (L, C, Element (C) + 1);

      Next (C);
   end loop;
end;
```

Если список содержит элементы более сложные, чем целые числа (например, управляемые типы), приведенный выше код не очень эффективен, так как вызов `function Element` вернет копию элемента. Чтобы избежать копирования, можно использовать вложенную подпрограмму и процедуры `Query_Element` и `Update_Element`, но это сделает код более сложным и менее читаемым.

Ada 2012 определяет три формы итераторов. Первая форма называется обобщенным итератором. Синтаксис и семантика для него приведены в справочном руководстве Ada 2012 (Reference Manual - 5.5.2), но вот пример его использования:

```
for C in L.Iterate loop
   Put_Line (Integer'Image (Element (C)));
   Replace_Element (L, C, Element (C) + 1);
end loop;
```

В этом коде C также является курсором, как и раньше, но принимает значения, возвращаемые функцией `Iterate`, определенной в экземпляре списка. Этот код в основном скрывает вызовы `First`, `Has_Element` и `Next`.

Вторая форма итератора, называемая итератором элемента контейнера, еще более удобна для пользователя и полностью скрывает использование курсоров. Эта форма итератора дает прямой доступ к элементам контейнера (см. Приложение A RM Ada для спецификаций предопределенных контейнеров -see Annex A of the Ada RM for specifications of the predefined containers):

```
for E of L loop    -- Note "of" instead of "in" here
   Put_Line (Integer'Image (E));
   E := E + 1;
end loop;
```

Третья форма итератора, называемая итератором компонента массива, аналогична итератору элемента контейнера, но применяется к типам массива. Вот пример такой формы:

```
declare
   Arr : array (1 .. 2) of Integer := (1 => 10, 2 => 20);

begin
   for E of Arr loop
       Put_Line (Integer'Image (E));
       E := E + 1; -- Change in place
   end loop;
end;
```

Как показано в примере, мы можем даже изменить элемент итератора `E` напрямую, и это изменяет значение в самом списке. Это также эффективный код, когда список содержит сложные типы, так как `E` не копия элемента, а ссылка на него.

Во второй части этой серии Gem объясняется, как писать собственные итераторы и как компилятор расширяет циклы, показанные выше.

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: Emmanuel Briot

AdaCore

Эммануэль Брио работает в AdaCore с 1998 года. Он принимал участие в различных проектах, в частности, ориентированных на графические пользовательские интерфейсы, включая GtkAda, GPS,

XML / Ada, GnatTracker и нашу внутреннюю CRM. Он получил степень инженера в Национальной школе телекоммуникаций - Брест, Франция (Ecole Nationale des Telecommunications - Brest, France).

Last Updated: 10/13/2017
Posted on: 6/11/2012

Обсуждение…